Nahuel Devesa Gil

Entrega Tarea 3.1, 3.2, 3.3, 3.4, 3.5 y 3.6

NDGt03e01-03:

3.1. Toma el proyecto del ejercicio 2.3 del tema anterior y desarrolla una clase de tipo @Controller que
contenga diferentes @GetMapping con las rutas quieras que devuelvan las vistas solicitadas (index,
palmares, galería-fotos, enlaces-externos).
Contesta en el PDF las siguientes cuestiones:
a) ¿Tienes que cambiar de ubicación las vistas? ¿Por qué?
b) ¿Tienes que cambiar el código HTML del menú de navegación de las páginas?
c) ¿Tienen que llamarse igual las rutas del GetMapping y las vistas?
La página index será servida para las URL: /index, /home, o simplemente /. Ya que las rutas y las vistas
no tienen por qué llamarse igual, renombra las vistas con el sufijo “view”: indexView.html, palmaresView.html, photogalleryView.html, linksView.html, etc. Así podemos distinguir bien por el
propio nombre lo que es una vista y lo que es una ruta o URL gestionada por el controlador.
Recuerda añadir en el application.properties la propiedad: spring.thymeleaf.cache=false y recuerda
también de que la etiqueta <html> lleva un atributo xmlns.
Utiliza th:href y th:src en lugar de los atributos HTML href y src respectivamente.
Elimina el mapping para los enlaces-externos y haz que muestre la vista linksView.html mediante un
archivo de configuración. Debes crear una clase que implemente WebMvcConfigurer, puedes llamarle
como quieras: WebMvcConfigurerImpl, WebMvcConfig, etc.
3.2. Añade al proyecto anterior contenido dinámico pasándole información a las plantillas mediante
un model y representándolo con etiquetas Thymeleaf. La página de inicio puede tener el año actual,
por ejemplo ©2024 tomado de la fecha del sistema del servidor. Para ello puedes usar el método
estático LocalDate.now().
La página de palmarés puede recibir la lista con los nombres de los títulos obtenidos por el equipo
(por ahora será un ArrayList de String en el controlador, pero más adelante debería tomar los datos
desde una base de datos).
3.3. Haz una copia del proyecto anterior y realiza los siguientes cambios:
• Si en la página de inicio, en la URL, se le pasa el parámetro: ?usuario=XXX mostrará el mensaje de
bienvenida con un texto personalizado para ese usuario, pero si no le pasa nada, será un mensaje
genérico (Bienvenido XXX a nuestra web vs. Bienvenido a nuestra web). Hazlo primero sin Optional,

luego ponla entre comentarios y haz una segunda versión con Optional.
• Añade BootStrap en su versión agnóstica (esto es, se define la versión empleada en el pom.xml
mediante webjars-locator).
• Utiliza fragmentos para no tener duplicado el código html tanto del <head> como del menú. El
menú podría utilizar la etiqueta <nav> y ser algo así:

```
<nav th:fragment="menu" class="navbar navbar-expand-sm">
 <ul class="navbar-nav">
 <li><a class="nav-link active" th:href="@{/}">Home</a></li>
 <li><a class="nav-link active" th:href="@{/palmares}">Palmarés</a></li>
 <li><a class="nav-link active" th:href="@{/galeria-fotos}">Galería de fotos</a></li>
 <li><a class="nav-link active" th:href="@{/enlaces}">Enlaces externos</a></li>
 </ul>
 </nav>
```

Es buena práctica que las clases estén agrupadas en carpetas (paquetes) por lo que podrías hacer una
carpeta ‘controllers’ para los controladores y otra ‘config’ para la clase de configuración creada
previamente. A medida que avance el curso tendremos nuevas carpetas: services, repositories, utils,
security, etc

a) ¿Tienes que cambiar de ubicación las vistas? ¿Por qué?

Sí, es posible que debas cambiar la ubicación de las vistas.

En un proyecto de Spring Boot, las vistas (archivos HTML) normalmente se colocan en el directorio src/main/resources/templates.
Si se decide organizar las vistas en subdirectorios dentro de templates, como user, admin, etc., es crucial que la ubicación coincida con la lógica de enrutamiento definida en los controladores de la aplicación. Esto ayuda a mantener una estructura clara y modular, facilitando el mantenimiento y la escalabilidad del proyecto.

b) ¿Tienes que cambiar el código HTML del menú de navegación de las páginas?

Sí, puede ser necesario cambiar el código HTML del menú de navegación.

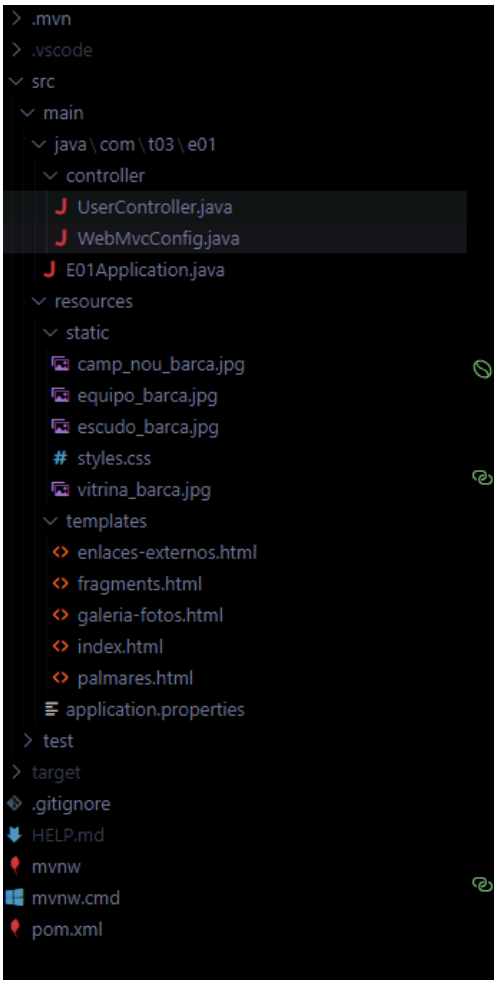
Si se utiliza un sistema de fragmentos en Thymeleaf para evitar la duplicación del código, el menú de navegación debería estar definido en un fragmento separado (por ejemplo, fragments.html), y cada página debe incluir este fragmento mediante la etiqueta th:replace.
Además, si se realizan cambios en la estructura de las rutas (por ejemplo, al agregar o renombrar rutas), también será necesario actualizar los enlaces (th:href) en el menú de navegación para que apunten a las nuevas rutas.

c) ¿Tienen que llamarse igual las rutas del GetMapping y las vistas?

No necesariamente, pero es recomendable mantener una convención clara.

Las rutas definidas en los métodos @GetMapping en los controladores de Spring Boot no tienen que llamarse exactamente igual que los nombres de las vistas (archivos HTML) en el directorio templates.
Sin embargo, es una buena práctica que el nombre de la ruta y el nombre de la vista coincidan o tengan una relación lógica, ya que esto facilita la comprensión del código y ayuda a evitar confusiones. Por ejemplo, si tienes un @GetMapping("/user/index"), lo más intuitivo sería que la vista se llame index.html y se ubique en el directorio correspondiente.

Para mi lo más difícil de estos 3 ejercicios fue el hecho de entender cómo funcionaba esta tecnología y los errores comunes de dicha tecnología.
El html quedaría similar a lo que vimos en el ejercicio 2.3 del tema anterior, la unica diferencia es como está planteado:



```java
@Controller
@RequestMapping("/user")
public class UserController {

    @GetMapping({"/", "/index", "/home"})
    public String index(@RequestParam(required = false) String usuario, Model model) {
    String mensajeBienvenida;

    if (usuario != null && !usuario.isEmpty()) {
        mensajeBienvenida = "Bienvenido " + usuario + " a nuestra web";
    } else {
        mensajeBienvenida = "Bienvenido a nuestra web";
    }

    model.addAttribute("mensajeBienvenida", mensajeBienvenida);
    model.addAttribute("year", LocalDate.now().getYear());
    return "index";
}

    @GetMapping("/palmares")
    public String palmares(Model model) {
        List<String> titulos = new ArrayList<>();
        titulos.add("Liga 2021");
        titulos.add("Copa del Rey 2020");
        titulos.add("Supercopa 2019");

        model.addAttribute("titulos", titulos);
        return "palmares";
    }

    @GetMapping("/galeria-fotos")
    public String galeria() {
        return "galeria-fotos";
    }

    @GetMapping("/enlaces")
    public String enlaces() {
        return "enlaces-externos";
    }

}
```

```java
package com.t03.e01.controller;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class WebMvcConfig implements WebMvcConfigurer {
    @Override
    public void addViewControllers(ViewControllerRegistry registry) {
        registry.addViewController("/enlaces-externos").setViewName("enlaces-externos");
    }
}
```

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <html xmlns:th="http://www.thymeleaf.org"></html>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" th:href="@{/webjars/bootstrap/5.3.0/css/bootstrap.min.css}">
    <link rel="stylesheet" th:href="@{/styles.css}">

    <title>FC Barcelona - Inicio</title>
</head>
<body>
    <header>
        <h1>FC Barcelona</h1>
        <nav th:replace="fragments :: menu"></nav>
    </header>
    <main>
        <h2 th:text="${mensajeBienvenida}"></h2>
        <p>El FC Barcelona, también conocido como "Barça", es un club de fútbol con sede en Barcelona, España. Fundado en 1899, ha sido un símbolo de la ciudad y una fuerza
        <img th:src="@{/escudo_barca.jpg}" alt="Logo del FC Barcelona">
    </main>
    <footer>
        <p>© <span th:text="${year}"></span> FC Barcelona Fans</p>
    </footer>
</body>
</html>
```

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" th:href="@{/webjars/bootstrap/5.3.0/css/bootstrap.min.css}">
    <link rel="stylesheet" th:href="@{/styles.css}">
    <html xmlns:th="http://www.thymeleaf.org"></html>
    <title>FC Barcelona - Palmarés</title>
</head>
<body>
    <header>
        <h1>FC Barcelona - Palmarés</h1>
        <nav th:replace="fragments :: menu"></nav>
    </header>
    <main>
        <h2>Lista de Títulos</h2>
        <ul>
            <li>La Liga: 26 títulos</li>
            <li>Copa del Rey: 31 títulos</li>
            <li>Champions League: 5 títulos</li>
            <li>Supercopa de España: 13 títulos</li>
            <li>Mundial de Clubes: 3 títulos</li>
        </ul>
        <h2>Ultimos titulos:</h2>
        <ul>
            <li th:each="titulo : ${titulos}" th:text="${titulo}"></li>
        </ul>

        <div class="imagen-vitrina"></div>
            <img th:src="@{/vitrina_barca.jpg}" alt="Vitrina de trofeos del FC Barcelona">
        </div>
    </main>
    <footer>
        <p>© 2024 FC Barcelona Fans</p>
    </footer>
</body>
</html>
```

```html
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" th:href="@{/webjars/bootstrap/5.3.0/css/bootstrap.min.css}">
    <link rel="stylesheet" th:href="@{/styles.css}">
    <html xmlns:th="http://www.thymeleaf.org">

    </html>
    <title>FC Barcelona - Enlaces Externos</title>
</head>

<body>
    <header>
        <h1>FC Barcelona - Enlaces Externos</h1>
        <nav th:replace="fragments :: menu"></nav>
    </header>
    <main>
        <h2>Enlaces de Interés</h2>
        <ul>

            <li><a href="https://www.fcbarcelona.com" target="_blank" rel="noopener noreferrer">Página Oficial del FC
                    Barcelona</a></li>
            <li><a href="https://es.wikipedia.org/wiki/FC_Barcelona" target="_blank" rel="noopener noreferrer">Wikipedia
                    del FC Barcelona</a></li>
            <li><a href="https://www.laliga.com/en-GB/clubs/barcelona" target="_blank" rel="noopener noreferrer">FC
                    Barcelona en La Liga</a></li>



        </ul>
    </main>
    <footer>
        <p>© 2024 FC Barcelona Fans</p>
    </footer>
</body>

</html>
```

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" th:href="@{/webjars/bootstrap/5.3.0/css/bootstrap.min.css}">
    <link rel="stylesheet" th:href="@{/styles.css}">
    <html xmlns:th="http://www.thymeleaf.org"></html>
    <title>FC Barcelona - Galería de Fotos</title>
</head>
<body>
    <header>
        <h1>FC Barcelona - Galería de Fotos</h1>
        <nav th:replace="fragments :: menu"></nav>
    </header>
    <main>
        <h2>Fotos del FC Barcelona</h2>
        <div class="imagenes">
            <img th:src="@{/camp_nou_barca.jpg}" alt="Camp Nou">
            <img th:src="@{/equipo_barca.jpg}" alt="Equipo del FC Barcelona">
        </div>
    </main>
    <footer>
        <p>© 2024 FC Barcelona Fans</p>
    </footer>
</body>
</html>
```

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Fragmentos de Thymeleaf</title>
    <link rel="stylesheet" th:href="@{/webjars/bootstrap/5.3.0/css/bootstrap.min.css}">
</head>
<body>
    <nav th:fragment="menu" class="navbar navbar-expand-sm" style="background-color: #9B1B30;">
        <div class="container-fluid justify-content-center">
            <ul class="navbar-nav text-center">
                <li class="nav-item"><a class="nav-link active" th:href="@{/user/index}" style="color: white;">Inicio</a></li>
                <li class="nav-item"><a class="nav-link active" th:href="@{/user/palmares}" style="color: white;">Palmarés</a></li>
                <li class="nav-item"><a class="nav-link active" th:href="@{/user/galeria-fotos}" style="color: white;">Galería de fotos</a></li>
                <li class="nav-item"><a class="nav-link active" th:href="@{/user/enlaces}" style="color: white;">Enlaces externos</a></li>
            </ul>
        </div>
    </nav>
</body>
</html>
```

El tiempo estimado para estos ejercicios era de 9 horas, al final lo realice en 7 horas 50 minutos (el tiempo que pase con javi me sirvió para resolver dos cosas que tenía mal en mi proyecto por eso cuento como horas trabajadas).
Me he dado cuenta que siempre intento hacer bastante vista hacia adelante por si acaso, intentaré ajustar mas los tiempos, aun así esta vez creo que medie muy bien mi tiempo.

NDGt03e04:

3.4. Implementa el ejemplo de los apuntes que genera números aleatorios en un nuevo proyecto.
Simplemente debes crear el controlador y la plantilla con el código mostrado.
Una vez que funcione correctamente, añade estilos CSS dinámicos. Los cambios a realizar serán los siguientes:
- Si la lista de números está vacía no se mostrará la tabla.
- Los números mayores de 50 se mostrarán con color de letra verde oscuro sobre fondo verde claro.
- Los números menores o iguales a 50 con color de letra rojo y fondo rosa.
Deberás crear las clases CSS que contengan los atributos de los dos puntos anteriores.

El ejercicio fue copiar el ejercicio de ejemplo, añadir algún detalle de css y alguna cosa para entenderlo mejor.

```java
@Controller
public class NumberController {

    private List<Integer> numbers = new ArrayList<>();

    @GetMapping("/list")
    public String getNumbers(Model model) {
        model.addAttribute("numbers", numbers);
        model.addAttribute("total", numbers.size());
        return "index";
    }

    @PostMapping("/add")
    public String addNumber() {
        Random random = new Random();
        numbers.add(random.nextInt(100) + 1);
        return "redirect:/list";
    }

    @GetMapping("/delete")
    public String deleteNumber(@RequestParam("index") int index) {
        if (index >= 0 && index < numbers.size()) {
            numbers.remove(index);
        }
        return "redirect:/list";
    }
}
```

```
    </style>
</head>

<body>
    <h1>Listado de numeros aleatorios</h1>

    <div th:if="${numbers.size() > 0}">
        <table>
            <thead>
                <tr>
                    <th>Número</th>
                    <th>Operación</th>
                </tr>
            </thead>
            <tbody>
                <tr th:each="number, iter : ${numbers}">
                    <td th:text="${number}" th:class="${number > 50} ? 'green' : 'red'"></td>
                    <td>
                        <a th:href="@{/delete(index=${iter.index})}">delete</a>
                    </td>
                </tr>
            </tbody>
        </table>
    </div>

    <p>Total números: <span th:text="${total}"></span></p>

    <form method="post" action="/add">
        <button type="submit">Nuevo número</button>
    </form>
</body>

</html>
```

Listado de numeros aleatorios

Número	Operación
86	delete
24	delete
43	delete
3	delete
43	delete
67	delete

Total números: 6

Nuevo número

NDGt03e05:

3.5. Realiza una aplicación con la apariencia que se muestra en la figura siguiente, de forma que presente tres imágenes a las que los visitantes pueden votar. Al clicar sobre cada una de las imágenes aumentará la cantidad de votos de esa imagen que se muestra debajo de ella. El contador de votos podrías mantenerlo en tres variables distintas, una para cada imagen, pero piensa que en un futuro añadamos más imágenes.

Puedes partir de esta plantilla HTML, eligiendo tú las imágenes de las películas que prefieras:

```
<!DOCTYPE html>
<html>
 <head>
 <meta charset="UTF-8">
 <meta http-equiv="X-UA-Compatible" content="IE=edge">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Elige tu película favorita</title>
 <link href="/webjars/bootstrap/css/bootstrap.min.css" rel="stylesheet">
 <script src="/webjars/bootstrap/js/bootstrap.bundle.min.js"></script>
 <link rel="stylesheet"
 href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.3.0/font/bootstrap-icons.css">
 <style> td { border: 1px solid #ddd;} </style>
 </head>
<body>
<h1>Elige tu película favorita</h1>
<table>
<tr>
 <td><a href="/voto?foto=0">
 <img src="images/avatar.jpg" alt="Avatar" style="width:200px;"></a></td>
 <td><a href="/voto?foto=1">
```

```
 <img src="images/cadenaPerpetua.jpg" alt="Cadena Perpetua"
style="width:200px;"></a></td>
 <td><a href="/voto?foto=2">
 <img src="images/pulpFiction.jpg" alt="Pulp Fiction" style="width:200px;"></a></td>
</tr>
<tr>
 <td><i class="bi bi-heart"></i><span>0</span></td>
 <td><i class="bi bi-heart"></i><span>0</span></td>
 <td><i class="bi bi-heart"></i><span>0</span></td>
</tr>
</table>
</body>
</html>
```

Por ahora estamos trabajando en memoria por lo que el contador de votos no se guardará cuando
cerremos la aplicación. En temas posteriores lo pasaremos a un repositorio persistente para arreglar
esta situación. Tampoco estamos controlando que un usuario vote varias veces a una misma película,
también solucionaremos esto más adelante.

El ejercicio fue bastante difícil ya que no tenía muy claro como hacer el controller, pero una vez que lo entendí avance a pasos agigantados.

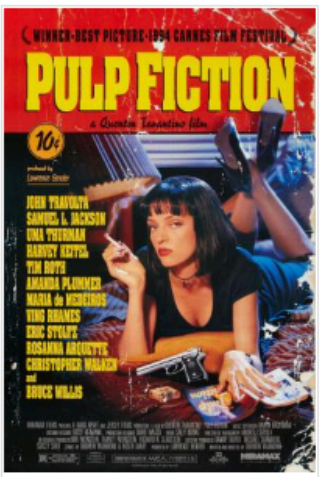
```java
@Controller
public class VotacionController {


    private int votosAvatar = 0;
    private int votosCadenaPerpetua = 0;
    private int votosPulpFiction = 0;

    @GetMapping("/")
    public String mostrarPeliculas(Model model) {

        model.addAttribute("votosAvatar", votosAvatar);
        model.addAttribute("votosCadenaPerpetua", votosCadenaPerpetua);
        model.addAttribute("votosPulpFiction", votosPulpFiction);
        return "votacion";
    }


    @GetMapping("/voto")
    public String votar(@RequestParam("foto") int foto) {
        switch (foto) {
            case 0:
                votosAvatar++;
                break;
            case 1:
                votosCadenaPerpetua++;
                break;
            case 2:
                votosPulpFiction++;
                break;
            default:
                break;
        }
        return "redirect:/";
    }
}
```

```
</head>
<body>
    <h1>Elige tu película favorita</h1>
    <table>
        <tr>
            <td>
                <a href="/voto?foto=0">
                    <img th:src="@{/imagenes/avatar.jpg}" alt="Avatar" style="width:200px;">
                </a>
            </td>
            <td>
                <a href="/voto?foto=1">
                    <img th:src="@{/imagenes/cadena_perpetua.jpg}" alt="Cadena Perpetua" style="width:200px;">
                </a>
            </td>
            <td>
                <a href="/voto?foto=2">
                    <img th:src="@{/imagenes/pulp_fiction.jpg}" alt="Pulp Fiction" style="width:200px;">
                </a>
            </td>
        </tr>
        <tr>
            <td>
                <i class="bi bi-heart"></i><span th:text="${votosAvatar}">0</span>
            </td>
            <td>
                <i class="bi bi-heart"></i><span th:text="${votosCadenaPerpetua}">0</span>
            </td>
            <td>
                <i class="bi bi-heart"></i><span th:text="${votosPulpFiction}">0</span>
            </td>
        </tr>
    </table>
</body>
</html>
```

Elige tu película favorita

♡5

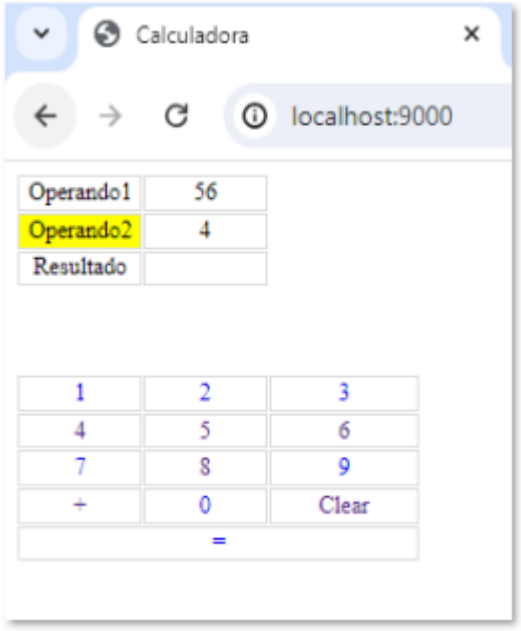
♡6



♡7

NDGt03e06:

3.6. Realiza una aplicación que implemente una calculadora como la mostrada en la figura siguiente.



El usuario iniciará introduciendo los dígitos del primer operando clicando en la botonera. Al pulsar en
el botón '+' pasará al segundo operando (si se pulsa ese botón en otra situación no hará nada). Luego
introducirá los dígitos del segundo operando y finalmente pulsará el botón '=' para mostrar el resultado
(si pulsa el botón'=' en otra situación, tampoco hará nada). El botón 'clear' vuelve a la situación inicial.
Te hará falta una variable de tipo enumeración para saber en qué estado estás: si introduciendo el
primero operando, el segundo o acabas de mostrar el resultado.
Si la vista se llama indexView.html y se llega a ella desde el mapping raíz: @GetMapping("/"), el resto de
mappings (añadir dígito, pulsar en el '+', etc.) lo más normal es que los métodos de esos mappings
terminen con: return "redirect:/" para volver a la vista inicial con sus datos en el Model.
Puedes partir de la siguiente plantilla HTML. Recuerda sustituir los atributos href de los enlaces, por
los equivalentes en Thymeleaf th:href.

```
<!DOCTYPE html>
<html>
 <head>
 <meta charset="UTF-8">
 <meta http-equiv="X-UA-Compatible" content="IE=edge">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Calculadora</title>
 <style>
```

```
 .focus {background-color: yellow;}
 a {text-decoration: none; width: 100%; padding-left: 2em; padding-right: 2em; }
 table td { border: 1px solid #ddd; width:5em;text-align: center; }
 </style>
 </head>
 <body>
 <table>
 <tr><td class="focus">Operando1</td><td></td></tr>
 <tr><td>Operando2</td><td></td></tr>
 <tr><td>Resultado</td><td></td> </tr>
 </table><br/> <br/> <br/>
 <table>
 <tr>
 <td><a href="/digito/1">1</a></td>
 <td><a href="/digito/2">2</a></td>
 <td><a href="/digito/3">3</a></td>
 </tr>
 <tr>
 <td><a href="/digito/4">4</a></td>
 <td><a href="/digito/5">5</a></td>
 <td><a href="/digito/6}">6</a></td>
 </tr>
 <tr>
 <td><a href="/digito/7">7</a></td>
 <td><a href="/digito/8">8</a></td>
 <td><a href="/digito/9">9</a></td>
 </tr>
 <tr>
 <td><a href="/suma">+</a></td>
 <td><a href="/digito/0">0</a></td>
 <td><a href="/clear">Clear</a></td>
 </tr>
 <tr>
 <td colspan="3"><a href="/igual">=</a></td>
 </tr>
 </table>
</body>
</html>
```

Una vez realice el ejercicio anterior entendí de una forma bastante “fácil” de hacer estos ejercicios aunque me costó un poco realizar el cambio de operador.

Operando1	4411
Operando2	788
Resultado	5199

1	2	3
4	5	6
7	8	9
+	0	Clear
=		

```
<body>


    <table>
        <tr>
            <td th:class="${estado.name() == 'OPERANDO1' ? 'focus' : ''}">Operando1</td>
            <td th:text="${operando1}"></td>
        </tr>
        <tr>
            <td th:class="${estado.name() == 'OPERANDO2' ? 'focus' : ''}">Operando2</td>
            <td th:text="${operando2}"></td>
        </tr>
        <tr>
            <td>Resultado</td>
            <td th:text="${resultado}"></td>
        </tr>
    </table><br /><br /><br />

    <table>
        <tr>
            <td><a th:href="@{/digito/1}">1</a></td>
            <td><a th:href="@{/digito/2}">2</a></td>
            <td><a th:href="@{/digito/3}">3</a></td>
        </tr>
        <tr>
            <td><a th:href="@{/digito/4}">4</a></td>
            <td><a th:href="@{/digito/5}">5</a></td>
            <td><a th:href="@{/digito/6}">6</a></td>
        </tr>
        <tr>
            <td><a th:href="@{/digito/7}">7</a></td>
            <td><a th:href="@{/digito/8}">8</a></td>
            <td><a th:href="@{/digito/9}">9</a></td>
        </tr>
        <tr>
            <td><a th:href="@{/suma}">+</a></td>
            <td><a th:href="@{/digito/0}">0</a></td>
            <td><a th:href="@{/clear}">Clear</a></td>
        </tr>
        <tr>
            <td colspan="3"><a th:href="@{/igual}">=</a></td>
        </tr>
    </table>

</body>
```

```java
@Controller
public class CalculadoraController {

    enum Estado {
        OPERANDO1, OPERANDO2, RESULTADO
    }

    private int operando1 = 0;
    private int operando2 = 0;
    private int resultado = 0;
    private Estado estado = Estado.OPERANDO1;

    @GetMapping("/")
    public String index(Model model) {
        model.addAttribute("operando1", operando1);
        model.addAttribute("operando2", operando2);
        model.addAttribute("resultado", estado == Estado.RESULTADO ? resultado : "");
        model.addAttribute("estado", estado);
        return "index";
    }

    @RequestMapping("/digito/{num}")
    public String addDigito(@PathVariable("num") int num) {
        if (estado == Estado.OPERANDO1) {
            operando1 = operando1 * 10 + num;
        } else if (estado == Estado.OPERANDO2) {
            operando2 = operando2 * 10 + num;
        }
        return "redirect:/";
    }

    @GetMapping("/suma")
    public String sumar() {
        if (estado == Estado.OPERANDO1) {
            estado = Estado.OPERANDO2;
        }
        return "redirect:/";
    }

    @GetMapping("/igual")
    public String calcularResultado() {
        if (estado == Estado.OPERANDO2) {
            resultado = operando1 + operando2;
            estado = Estado.RESULTADO;
        }
        return "redirect:/";
    }

    @GetMapping("/clear")
    public String clear() {
        operando1 = 0;
        operando2 = 0;
        resultado = 0;
        estado = Estado.OPERANDO1;
        return "redirect:/";
    }
}
```

El tiempo estimado fue de 4 horas del 3.4 al 3.6.
Llevándome un total de 3 horas y poco (como tengo empresas convalidada use esta hora para avanzar).