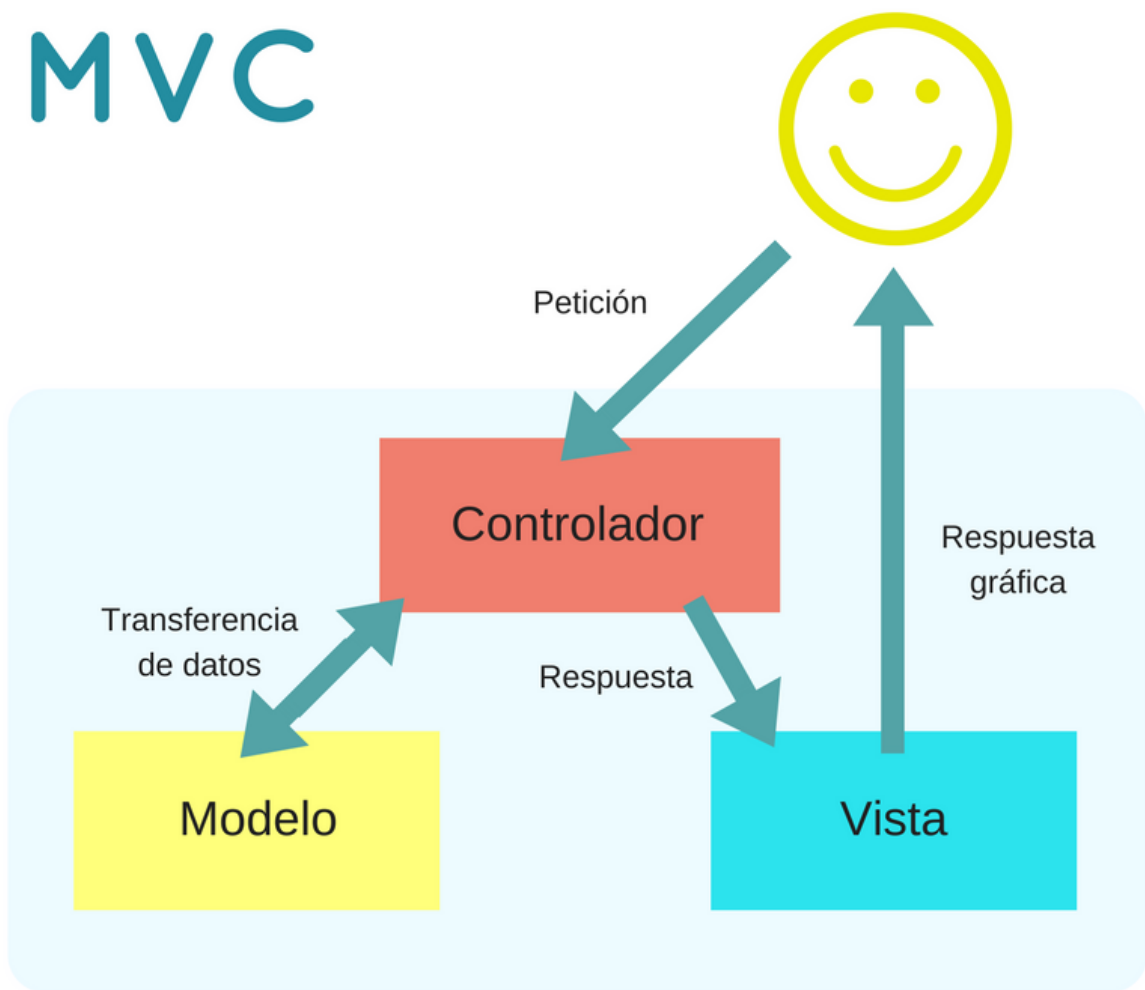


Tema 3: Controladores y Vistas

MVC



Índice

Tema 3: Controladores y Vistas	1
Índice	2
Ejercicios	3
3.1.	3
3.2.	4
3.3.	6
3.4.	7
3.5.	8

Ejercicios

3.1.

Toma el proyecto del ejercicio 2.4 del tema anterior y desarrolla una clase controladora que contenga diferentes `@GetMapping` que devuelvan las vistas solicitadas (index, quienes-somos, productos, contacta) .

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

Si, porque antes las rutas se controlaban mediante links y ahora tenemos controladores que son los encargados de gestionarlas.

Las pasamos a templates.

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

No es necesario.

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

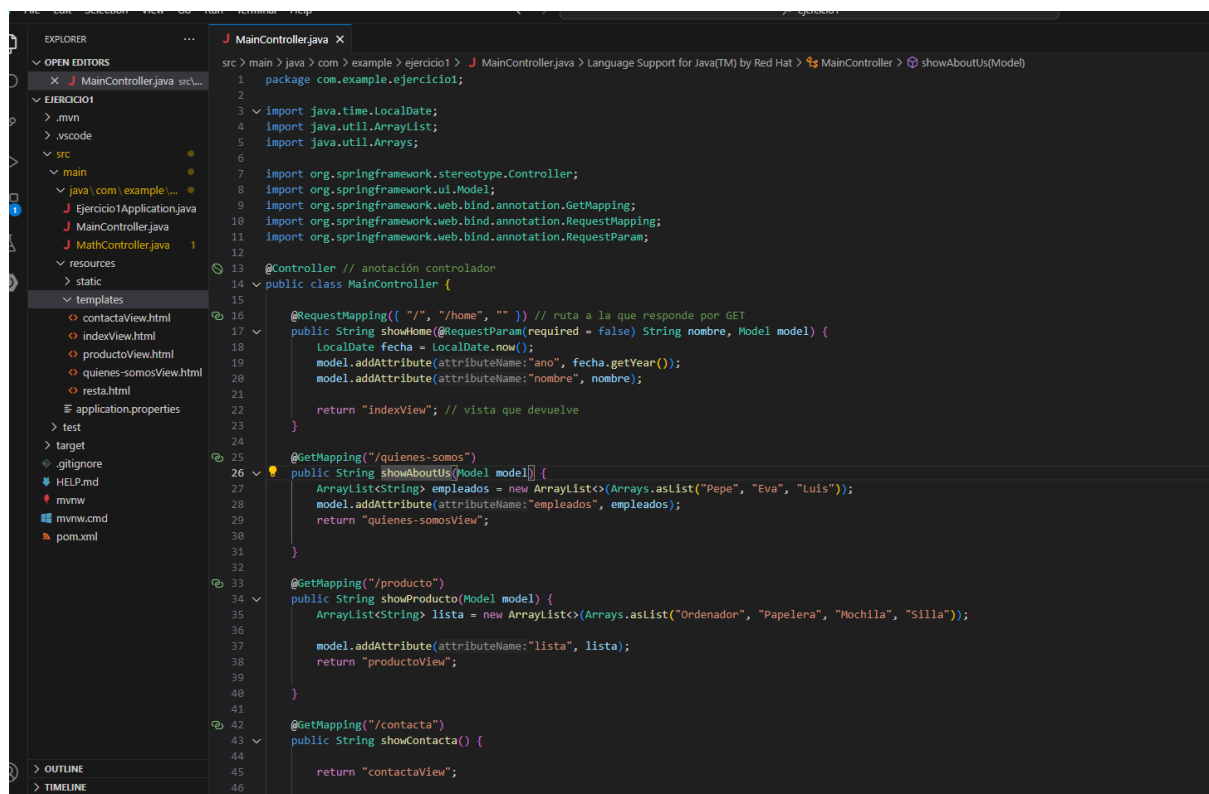
No, ya que es el controlador la que se encarga de en función de una ruta dada redirigir a una vista.

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”: `homeView.html`, `aboutUsView.html`, `productsView.html`, `contactView.html`.

(Opcional) Elimina el mapping `quienes-somos` y haz que muestre la vista `aboutUsView.html` mediante

un archivo de configuración (implementando `WebMvcConfigurer`).



```
1 package com.example.ejercicio1;
2
3 import java.time.LocalDate;
4 import java.util.ArrayList;
5 import java.util.Arrays;
6
7 import org.springframework.stereotype.Controller;
8 import org.springframework.ui.Model;
9 import org.springframework.web.bind.annotation.GetMapping;
10 import org.springframework.web.bind.annotation.RequestMapping;
11 import org.springframework.web.bind.annotation.RequestParam;
12
13 @Controller // anotación controlador
14 public class MainController {
15
16     @RequestMapping({ "/", "/home", "" }) // ruta a la que responde por GET
17     public String showHome(@RequestParam(required = false) String nombre, Model model) {
18         LocalDate fecha = LocalDate.now();
19         model.addAttribute(attributeName:"ano", fecha.getYear());
20         model.addAttribute(attributeName:"nombre", nombre);
21
22         return "indexView"; // vista que devuelve
23     }
24
25     @GetMapping("/quienes-somos")
26     public String showAboutUs(Model model) {
27         ArrayList<String> empleados = new ArrayList<>(Arrays.asList("Pepé", "Eva", "Luis"));
28         model.addAttribute(attributeName:"empleados", empleados);
29         return "quienes-somosView";
30     }
31
32     @GetMapping("/producto")
33     public String showProducto(Model model) {
34         ArrayList<String> lista = new ArrayList<>(Arrays.asList("Ordenador", "Papelera", "Mochila", "Silla"));
35
36         model.addAttribute(attributeName:"lista", lista);
37         return "productoView";
38     }
39
40     @GetMapping("/contacta")
41     public String showContacta() {
42
43         return "contactaView";
44     }
45
46 }
```

Creo cada controlador para manejar cada vista, así se vuelve independiente el nombre de las vistas con las rutas asociadas.

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 ©2023 tomado de la fecha del sistema del servidor. La página de productos puede recibir la lista con los nombres de los productos que ofrece (por ahora puede ser un arraylist de String en el controlador, pero más adelante tomará los datos de la base de datos).

```

11
12 @Controller // anotación controlador
13 public class MainController {
14
15     @GetMapping({ "/", "/home", "" }) // ruta a la que responde por GET
16     public String showHome(@RequestParam(required = false) String nombre, Model model) {
17         LocalDate fecha = LocalDate.now();
18         model.addAttribute(attributeName:"ano", fecha.getYear());
19         model.addAttribute(attributeName:"nombre", nombre);
20
21         return "indexView"; // vista que devuelve
22     }
23

```

Creo un GetMapping que controla las principales rutas y que capture de forma dinámica la fecha de hoy y se lo pase a la vista para que salga por pantalla.

```

48
49     <li><a href="/">Inicio</a></li>
50     <li><a href="/quienes-somos">Quiénes Somos</a></li>
51     <li><a href="/producto">Productos</a></li>
52     <li><a href="/contacta">Contacta</a></li>
53     </ul>
54 </nav>
55 <div class="container">
56     <h2>Página de Inicio</h2>
57     <p>Bienvenido a nuestro sitio web <span th:text="${nombre}"></span><br>
58     <h2 th:text="${ano}"></h2>
59 </div>
60 
61 </body>
62 </html>
63

```

Introduzco un th:text para insertar el nombre que se me está pasando desde el controlador de forma dinámica.

```

    @GetMapping("/producto")
    public String showProducto(Model model) {
        ArrayList<String> lista = new ArrayList<>(Arrays.asList("Ordenador", "Papelera", "Mochila", "Silla"));

        model.addAttribute(attributeName:"lista", lista);
        return "productoView";
    }

    @GetMapping("/contacta")
    public String showContacta() {

```

Creo un controlador para mi pagina de productos que pase una ArrayList a la vista.

```

        <li><a href="/producto">Productos</a></li>
        <li><a href="/contacta">Contacta</a></li>
    </ul>
</nav>
<div class="container">
    <ul th:each="elemento:${lista}">
        <li th:text="${elemento}" ></li>
    </ul>
</div>
</body>
</html>

```

Mediante th:each itero cada elemento de la lista y le asigno el nombre elemento para después crear li de forma dinámica.

3.3.

Haz una copia del proyecto anterior y sobre él: si en la página de inicio, 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).

```

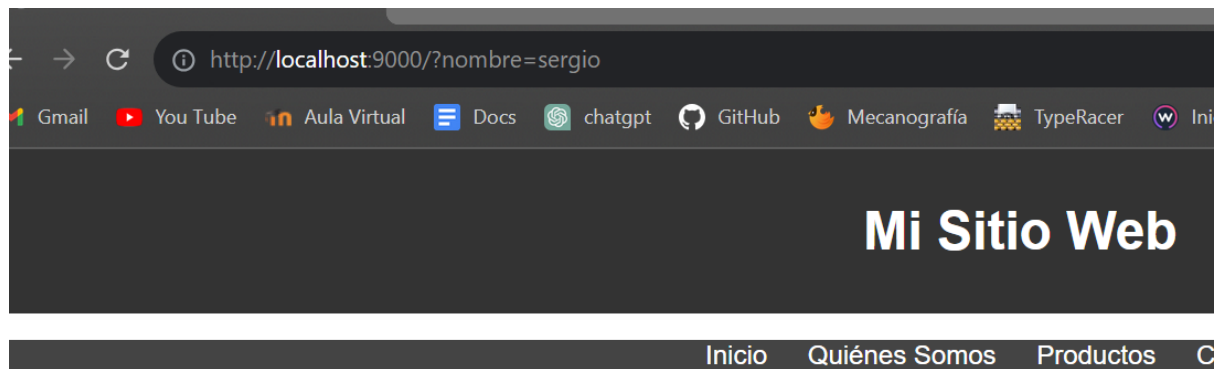
@Controller // anotación controlador
public class MainController {

    @GetMapping({ "/", "/home", "" }) // ruta a la que responde por GET
    public String showHome(@RequestParam(required = false) String nombre, Model model) {
        LocalDate fecha = LocalDate.now();
        model.addAttribute(attributeName:"ano", fecha.getYear());
        model.addAttribute(attributeName:"nombre", nombre);

        return "indexView"; // vista que devuelve
    }
}

```

Con requestParam y definiendo que el parámetro es opcional hacemos que si el usuario le pasa por parámetro el valor de nombre salga un mensaje personalizado para así saludarle.



Página de Inicio

Bienvenido a nuestro sitio web sergio.
Aquí encontrarás información sobre nuestros productos y servicios.

Como podemos ver se efectúa correctamente.

3.4.

Implementa el ejemplo de los apuntes que generan números aleatorios en un nuevo proyecto.

Simplemente debes crear el controlador y la plantilla con el código mostrado.

```
src > main > java > com > example > ejercicio4 > HomeController.java > Language Support for Java(TM) by Red Hat > {} com.example.ejercicio4

4  import java.util.Arrays;
5  import java.util.LinkedHashSet;
6  import java.util.Random;
7  import java.util.Set;
8
9  import org.springframework.stereotype.Controller;
10 import org.springframework.ui.Model;
11 import org.springframework.web.bind.annotation.GetMapping;
12 import org.springframework.web.bind.annotation.PathVariable;
13
14 @Controller
15 public class HomeController {
16     Random random = new Random();
17     public Set<Integer> lista = new LinkedHashSet<>();
18
19     @GetMapping("/{}/list")
20     public String showNumber(Model model){
21         model.addAttribute(attributeName:"cantidadTotal", lista.size());
22         model.addAttribute(attributeName:"listaNumeros", lista);
23         return "index";
24     }
25
26
27
28     @GetMapping("/new")
29     public String addNumber(){
30         boolean añadido;
31         do {
32             añadido = lista.add(random.nextInt(100)+1);
33         } while (!añadido);
34         return "redirect:/list";
35     }
36
37
38
39
40     @GetMapping("/delete/{id}")
41     public String deleteNumber(@PathVariable Integer id){
42         lista.remove(id);
43         return "redirect:/list";
44     }
45
46 }
47
```

Creo el controlador.

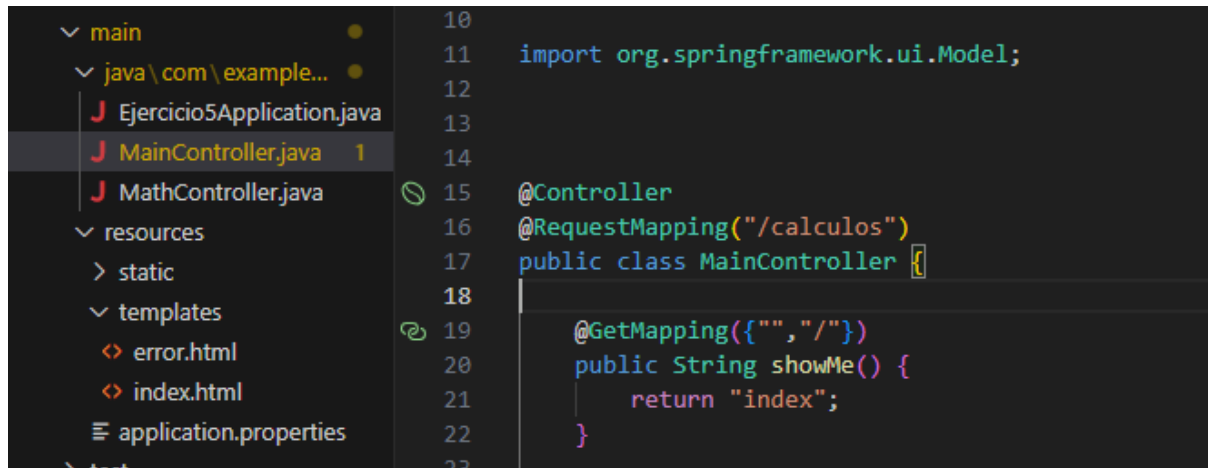


Funciona correctamente.

3.5.

Vuelve al proyecto del ejercicio 2.2 para trabajar con el pase de parámetros a los controladores en el path. Vamos a crear un nuevo controlador, con una nueva ruta base llamada /calculos que deberá realizar las siguientes tareas:

- Pasa la página index.html a la carpeta /templates y crea un controlador para ella.



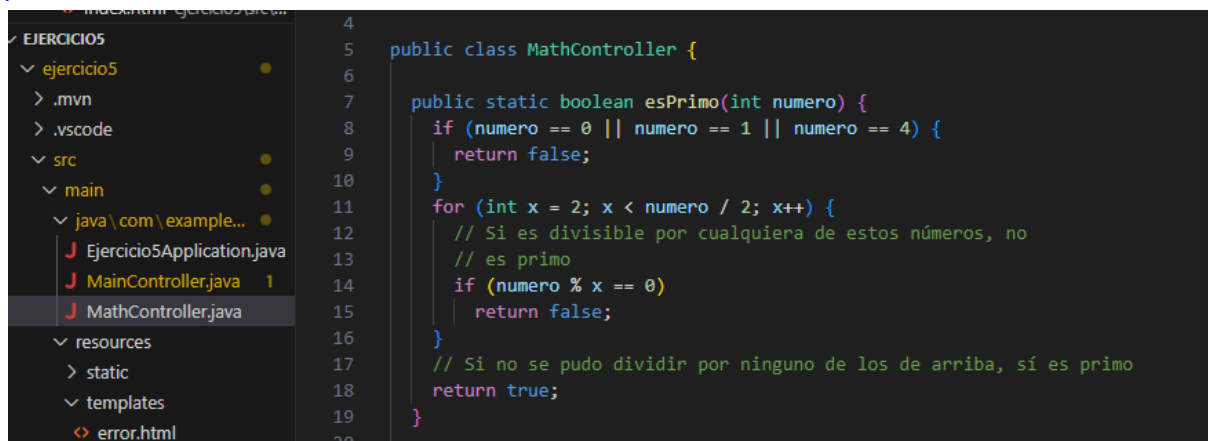
```
10
11 import org.springframework.ui.Model;
12
13
14
15 @Controller
16 @RequestMapping("/calculos")
17 public class MainController {
18
19     @GetMapping({"/", "/"})
20     public String showMe() {
21         return "index";
22     }
23
```

Creo un RequestMapping para fijar la ruta inicial que va a ser “/calculos” y también creo un GetMapping para la ruta del index.

b) Mediante la URL: /calculos/primo?numero=X devolverá una página diciendo que el número X

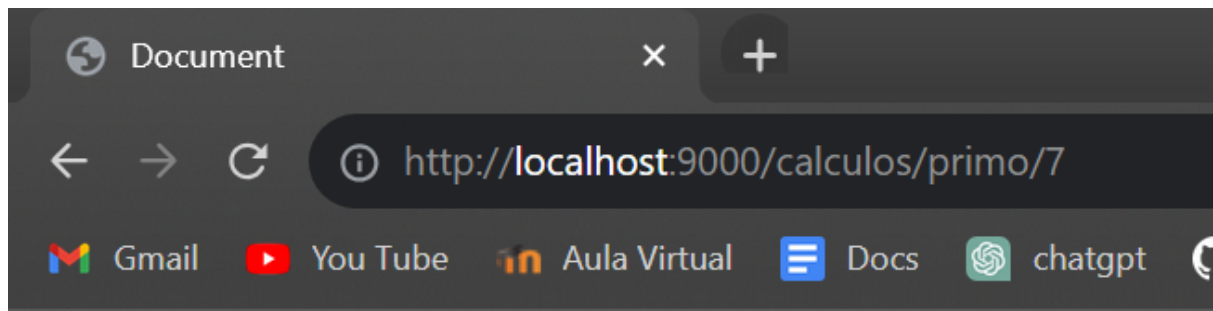
es primo o no. Por ahora, haz los cálculos en un método del propio controlador, aunque en capítulos siguientes los moveremos a otras capas.

Creo un GetMapping que mediante Path se introduce el numero que queremos ver si es primo o no.



```
4
5 public class MathController {
6
7     public static boolean esPrimo(int numero) {
8         if (numero == 0 || numero == 1 || numero == 4) {
9             return false;
10        }
11        for (int x = 2; x < numero / 2; x++) {
12            // Si es divisible por cualquiera de estos números, no
13            // es primo
14            if (numero % x == 0)
15                return false;
16        }
17        // Si no se pudo dividir por ninguno de los de arriba, sí es primo
18        return true;
19    }
20
```

Dentro del GetMapping uso una funcion que me dice si es primo que he creado en una clase aparte para usar funciones especificas.



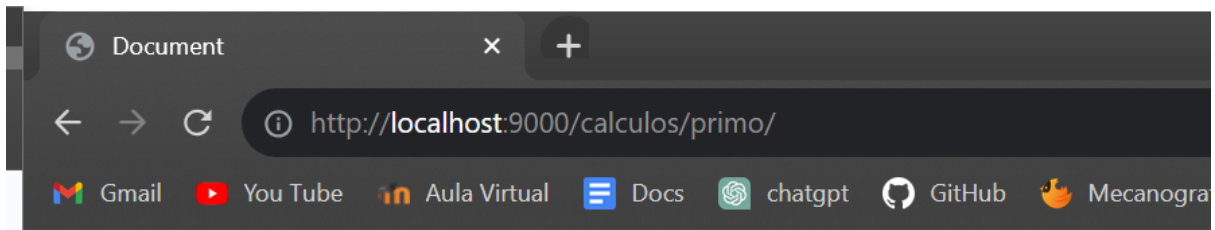
Calculos

El numero es primo

Funciona correctamente.

c) Modifica el apartado anterior, para que, si no introduce ningún número al que calcular si es primo o no, lo redirija a un controlador que trate el error. Este controlador por ahora simplemente mostrará la página de inicio.

```
@GetMapping("/primo/")
public String showError(){
    return "error";
}
```



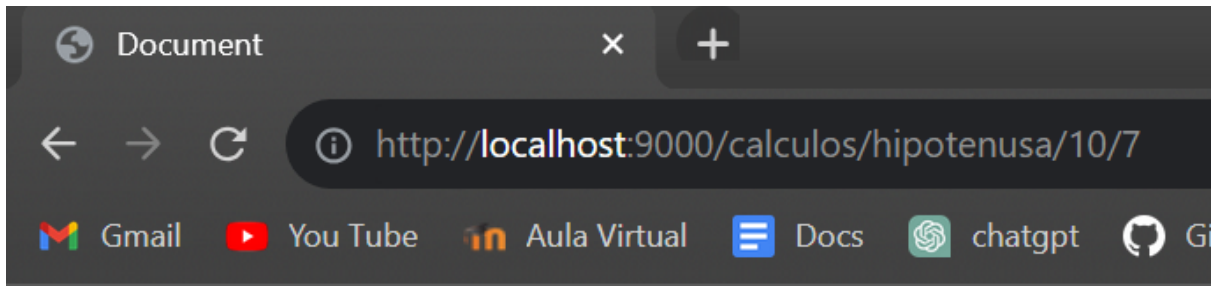
Error, has introducido mal los numeros

He creado una página que sea de error y cuando se introduce el primo sin ningún parámetro se redirige a ella.

d) Mediante la URL /calculos/hipotenusa/X/Y devolverá una página con el valor de la hipotenusa correspondiente a los catetos X e Y. Si los números son negativos deberá redirigir al controlador que trate el error.

```
@GetMapping("/hipotenusa/{x}/{y}")
public String calculaHipotenusa(@PathVariable String x, @PathVariable String y, Model model){
    String respuesta = String.valueOf(Math.hypot(Double.parseDouble(x), Double.parseDouble(y)));
    model.addAttribute(attributeName:"respuesta", "La hipotenusa de "+y+" y "+ x +" es " + respuesta);
    if(Integer.parseInt(x) < 0 || Integer.parseInt(y) < 0) {
        return "error";
    }else{
        return "index";
    }
}
```

Creo el GetMapping para que en caso de que sean menor que 0 me dirija a la página de error y si no que devuelva la hipotenusa .



Calculos

La hipotenusa de 7 y 10 es 12.206555615733702

[Funciona correctamente.](#)

e) Mediante la URL /calculos/sinRepetidos/X devolverá una página con X números aleatorios comprendidos entre 1 y 100, sin repetidos y ordenados ascendentemente. Obviamente X debe ser un número entero y estar comprendido entre 1 y 100, sino redirigirá a la página de error. Los mostrará en una tabla con una sola columna y cada número en una fila.
Pregunta: ¿Qué colección de Java es la más adecuada para que no contenga repetidos y estén ordenados?

```
@GetMapping("/sinRepetidos/{x}")
public String numeroAleatorio(@PathVariable Integer x, Model model){

    if(x >= 1 && x <= 100){
        model.addAttribute(attributeName: "numeros", MathController.generaAleatorios(x));
        System.out.println(MathController.generaAleatorios(x));
        return "index";
    }else{
        return "error";
    }
}
```

Creo un GetMapping en el que use la Clase MathController y el método general Aleatorio.

```
public static ArrayList<Integer> generaAleatorios(int numero) {  
    ArrayList<Integer> numeros = new ArrayList<>();  
    int nuevoNumero;  
  
    for (int i = 0; i < numero; i++) {  
        do {  
            nuevoNumero = (int) (Math.random() * 100 + 1);  
        } while (numeros.contains(nuevoNumero));  
        numeros.add(nuevoNumero);  
    }  
    return numeros;  
}
```

Pese a haber utilizado una ArrayList normal la mejor opción es usar un hasSet porque no tiene números repetidos.

f) Mediante la URL /calculos/divisores/X devolverá una lista con divisores del número X (cada número en un párrafo). Cada elemento de esa lista será un enlace, de forma que si el usuario clicca en uno de ellos mostrará los divisores del mismo. Ejemplo: para la URL /calculos/divisores/12 mostrará 1,2,3,6, 12 y podremos clicar por ejemplo en el 6 y mostrará 1,2,3,6 y así sucesivamente.

```
@GetMapping("/divisores/{x}")  
public String divisores(@PathVariable Integer x, Model model) {  
    model.addAttribute(attributeName: "divisores", MathController.generaDivisores(x));  
    return "index";  
}
```

GetMapping usando generarDivisores.

```
public static ArrayList<Integer> generaDivisores(int numero){  
    ArrayList<Integer> divisores = new ArrayList<>();  
    for(int i = 1 ; i<= numero ; i++){  
        if(numero%i==0){  
            divisores.add(i);  
        }  
    }  
    return divisores;  
}
```

Método que devuelve un arrayList con todos los divisores.

g) Haz una versión del ejercicio anterior, pero con el parámetro X en la query de la URL, por ejemplo: /calculos/divisores?num=X. También las URL generadas para los divisores serán enlaces con el parámetro en la query.

```
</ul>
<ul th:each="divisor:${divisores}">
  <li><a th:href="@{/calculos/divisores/{x}(x=${divisor})}" th:text="${divisor}"></a></li>
</ul>
/body>
</html>
```

Usando la etiqueta th:each itero todos los divisores y los convierto dinámicamente en enlaces en los que haces click llaman al metodo de calculos/ y su propio numero.

