

Nombre del alumno: Jefry André Cruz Yumán

Carné:	2021 644	Grado y Jornada	Quinto periodo matutino
Curso:	TIC's, Taller	Fecha de entrega:	02/09/2025
Nombre de la tarea:	Reporte de ahorcado		

Reporte de cómo se trabajó el ahorcado:

Partes del reporte

- Explicación de la estructura del jsp index.
- Explicación de la estructura del jsp ahorcado.
- Explicación del método controller para el control de vistas.
- Explicación de la formación del js de mi ahorcado.
- Explicación extra de como se agregaron las palabras y pistas con db.
- Explicación referente a como se realizo el login y el registrar.

Enlace de mi repositorio:

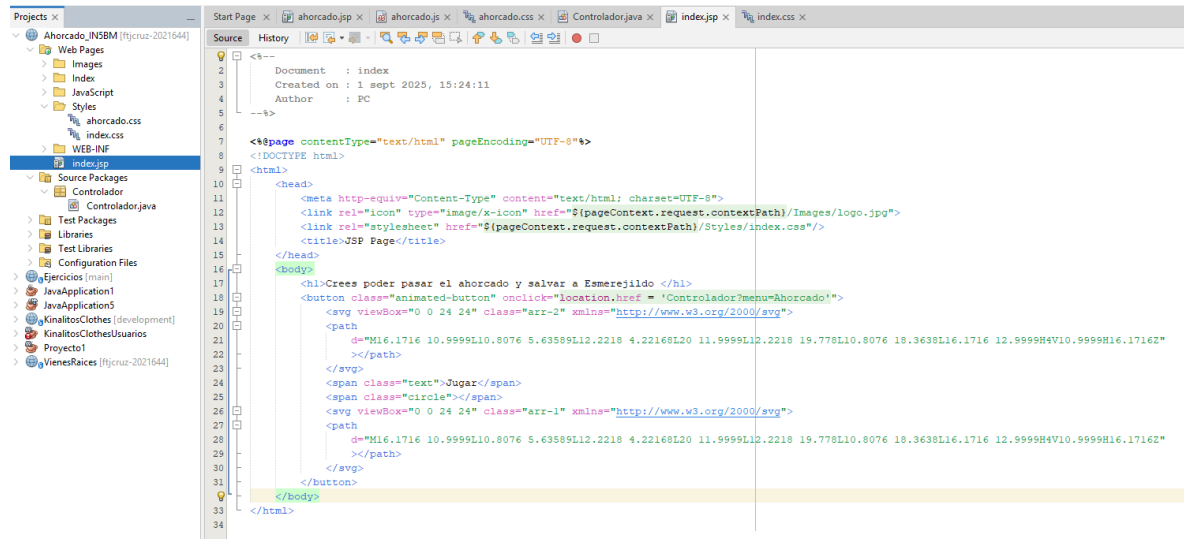
https://github.com/jcruz-2021644/Ahorcado_IN5BM.git

Enlace de mi trello:

<https://trello.com/invite/b/68b5ddd3ab1a3166943f4f41/ATTI6c7478246802d76f4427bc19f8fdafdfABCCDD99/ahorcado-in5bm>

Explicación de la estructura del jsp index

En este jsp yo implemente únicamente un fondo de gif y un botón el cual por medio del onclick me lleva a la vista del juego ahorcado también escribir un titulo de motivación para que se anime a jugar.



```

1  <!--
2  Document : index
3  Created on : 1 sept 2025, 15:24:11
4  Author : PC
5  -->
6
7  <!--
8  <!--page contentType="text/html" pageEncoding="UTF-8">
9  <!--
10 <html>
11
12 <head>
13 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14 <link rel="icon" type="image/x-icon" href="${pageContext.request.contextPath}/Images/logo.jpg">
15 <link rel="stylesheet" href="${pageContext.request.contextPath}/Styles/index.css"/>
16 <title>JSP Page</title>
17 </head>
18 <body>
19 <h1>Crees poder pasar el ahorcado y salvar a Esmerajildo </h1>
20 <button class="animated-button" onclick="location.href = 'Controlador?menu=Ahorcado'">
21 <svg viewBox="0 0 24 24" class="arr-2" xmlns="http://www.w3.org/2000/svg">
22 <path
23 d="M16.1716 10.9999L10.8076 5.63589L12.2218 4.22168L20 11.9999L12.2218 19.7781L10.8076 18.3638L16.1716 12.9999H4V10.9999H16.1716Z"
24 </path>
25 </svg>
26 <span class="text">Jugar</span>
27 <span class="circle"></span>
28 <svg viewBox="0 0 24 24" class="arr-1" xmlns="http://www.w3.org/2000/svg">
29 <path
30 d="M16.1716 10.9999L10.8076 5.63589L12.2218 4.22168L20 11.9999L12.2218 19.7781L10.8076 18.3638L16.1716 12.9999H4V10.9999H16.1716Z"
31 </path>
32 </svg>
33 </button>
34 </body>
35 </html>

```

```

<!--
Document : ahorcado
Created on : 1 sept 2025, 17:33:22
Author : PC
-->

<?page contentType="text/html" pageEncoding="UTF-8"?>
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Ahorcado</title>
<link rel="icon" type="image/x-icon" href="${pageContext.request.contextPath}/Images/logo.jpg">
<link rel="stylesheet" href="${pageContext.request.contextPath}/Styles/ahorcado.css"/>
</head>
<body>
<div class="contenedor-juego">
<div class="header">
<h1 class="titulo">AHORCADO</h1>
<div class="tiempo" id="tiempo-display">06:00</div>
</div>
<div class="contenedor-principal">
<div class="area-dibujo">
<!-- Aquí aparecerá la imagen del ahorcado cuando cometas errores -->
<img id="imagen-ahorcado" alt="Ahorcado">
</div>
<div class="seccion-abecedario">
<div class="botones-control">
<button class="button-acciones" id="btn-iniciar">Iniciar</button>
<button class="button-acciones" id="btn-reiniciar">Reiniciar</button>
<button class="button-acciones" id="btn-pausar">Pausar</button>
<button class="button-acciones" id="btn-salir">Salir</button>
</div>
<div class="contenedor-abecedario">
<div class="word-container">
<p id="palabra">¡Presiona Iniciar para comenzar!</p>
</div>
<div class="abecedario-tabla" id="teclado">
<!-- al final elimine mi listado porque los botones los tengo que crear en el for -->
</div>
</div>
</div>
<div class="seccion-abajo">
<div class="cuadro-intentos">
<div class="numero-intentos" id="intentos">6</div>
<span>INTENTOS</span>
</div>
<div class="rectangulo-pista">
<p id="pista">Pista: ¡Haz clic en "Iniciar" para comenzar el juego!</p>
<div id="mensaje"></div>
</div>
<div class="cuadro-imagen">
<img id="imagen" />
</div>
</div>
</div>
<script>
const contextPath = '${pageContext.request.contextPath}';
</script>
<script src="${pageContext.request.contextPath}/JavaScript/ahorcado.js"></script>
</body>
</html>

```

Explicación del método controller para el control de vistas

En el controlador solamente tenemos nuestro método `processRequest` donde nosotros tenemos nuestros cambios de vista en la cual se basa por medio de secuencias `if` para poder decir que si el menú es igual a Ahorcado o Index nos redirige a nuestra vista ejemplo: `Controlador?menu=Ahorcado`. Esta nos llevara a la vista `ahorado.jsp` porque el menú es igual a Ahorcado.

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException, ParseException {
    String menu = request.getParameter(name: "menu");

    if (menu.equals(anObject: "Ahorcado")) {
        request.getRequestDispatcher(path: "Index/ahorcado.jsp").forward(request, response);
    } else if (menu.equals(anObject: "Index")) {
        request.getRequestDispatcher(path: "index.jsp").forward(request, response);
    }
}
```

Explicación de la formación del js de mi ahorcado.

Para poder empezar con mi ahorcado lo primero que hice fue crear mi array con diferentes datos guardados referente a las palabras, pistas e imágenes de ayuda.

```
const palabras = [  
  {  
    palabra: "TORREFACTO",  
    pista: "Negro como la noche, en taza me encontrarás, si me pruebas con azúcar, ¿sabes cómo me llamarás?",  
    imagen: contextPath + "/Images/terrefacto.jpg"  
  },  
  {  
    palabra: "SEPTIEMBRE",  
    pista: "Entre el calor que se apaga y el frío que viene ligero, traigo la patria en bandera y otoño en mi sombrero.",  
    imagen: contextPath + "/Images/septiembre.jpg"  
  },  
  {  
    palabra: "MANZANILLA",  
    pista: "Soy una flor sencilla y pequeña, me buscan por mi sabor, en infusiones me toman para calmar el dolor.",  
    imagen: contextPath + "/Images/manzanilla.jpg"  
  },  
  {  
    palabra: "BOMBARDERO",  
    pista: "No soy ave, pero vuelo y en los cielos hago un estruendo, llevo bombas peligrosas cuando al combate me enciendo.",  
    imagen: contextPath + "/Images/bombardero.jpg"  
  },  
  {  
    palabra: "ABECEDARIO",  
    pista: "De la A a la Z me puedes recitar, con mis letras se construyen las palabras al hablar.",  
    imagen: contextPath + "/Images/abecedario.jpg"  
  }  
];
```

Luego mande a traer todos los id que necesito los cuales implemente en el ahorcado.jsp luego de tener todos mis elementos instancio todas mis variables del juego

```
const espacioPalabra = document.getElementById('palabra');
const pistaElemento = document.getElementById('pista');
const intentosElementos = document.getElementById('intentos');
const messageElement = document.getElementById('mensaje');
const contenedorTeclado = document.getElementById('teclado');
const tiempoDisplay = document.getElementById('tiempo-display');
const btnIniciar = document.getElementById('btn-iniciar');
const btnReiniciar = document.getElementById('btn-reiniciar');
const btnPausar = document.getElementById('btn-pausar');
const btnSalir = document.getElementById('btn-salir');
const imagenElement = document.getElementById('imagen');

// Variables del juego
// es la palabra a adivinar y lo dejamos vacio porque despues lo setemos en el random para la palabra
let palabraSecreta = '';
// el array de las letras que ya hemos usado
let letrasAdivinadas = [];
let intentos = 7;
// lo mismo la dejamos vacia depues lo seteamos
let pista = '';
// lo mismo la dejamos vacia depues lo seteamos
let imagenPalabra = '';
// este tiene que ser boolean porque dice que el juego esta activo
let juegoIniciado = false;
// lo dejamos null porque no hay ningun cronometro funcionndo por el momento
let temporizador = null;
// le dejamos el tiempo de 6 min porque serian 6 min de 60 seg
let tiempoRestante = 6 * 60;
// indica que es falso por eso termina
let juegoTerminado = false;
// lo mismo indica que es falso y no termina sino que se queda en pausa
let juegoPausado = false;
```

También tengo que crear un array con las imágenes de mi muñeco porque cree un array porque las imágenes del ahorcado se iran sobre poniendo para poder ir armando nuestro muñeco en total son 7 imágenes porque son 7 intentos que tiene la persona para fallar, también importamos nuestro elemento con el id imagen-ahorcado el cual nos servirá para establecer la imagen dentro de él.

```
// Las imagenes de los del ahorcado
const imagenesAhorcado = [
  contextPath + '/Images/ahorcado1.png',
  contextPath + '/Images/ahorcado2.png',
  contextPath + '/Images/ahorcado3.png',
  contextPath + '/Images/ahorcado4.png',
  contextPath + '/Images/ahorcado5.png',
  contextPath + '/Images/ahorcado6.png',
  contextPath + '/Images/ahorcado7.png'
];

// Elemento para mostrar la imagen del ahorcado
const imagenAhorcado = document.getElementById('imagen-ahorcado') || crearElementoImagenAhorcado();
```

Luego de esto creamos el un elemento que es el que nosotros usamos para setear la imagen dentro del área de dibujo esta parte de crearlo no es necesario ya que ya lo creamos en el jsp pero es bueno volverlo a crear porque se podría decir que se limpia cada vez que lo creamos luego de crearlo traemos el atributo img a el cual le agregamos el id de imagen-ahorcado también agregamos esta imagen al área dibujo para mostrar nuestro muñeco y retornamos la imagen para mostrarlo.

```
// Crear elemento para la imagen del ahorcado
function crearElementoImagenAhorcado() {
    //creamos la imagen
    const img = document.createElement('img');
    //le otorgamos el id de la imagen dl ahorcado para darcelo al mostrarimagen
    img.id = 'imagen-ahorcado';
    // Agregar la imagen al area de dibujo
    const areaDibujo = document.querySelector('.area-dibujo');
    if (areaDibujo) {
        //añadimos la imagen al area del dibujo
        areaDibujo.appendChild(img);
    }
    return img;
}
```

Con el tema del cronometro o temporizador empecé por el formatearTiempo este que hace como su nombre lo dice nos ayuda para que el cronometro se reinicie y como lo hace primero tenemos que instanciar una contante llamada mins que hace referencia a los minutos entonces decimos que usamos el math.floor para solo obtener los enteros no decimales de la división de pasar seg a mins luego tenemos otra constante de segundos que es el residuo de la división `seg%60` también le di un formato que seria el 00:00 ya que pasamos los mins a tipo String para poder implementarlos y le agregamos que tenga dos espacios y uno de ellos sea 0 ya se para los mins o los seg .

```
function formatearTiempo(segundos) {
    // pasamos de segundos a minutos y suamos el floor para no tener decimales solo enteros
    const mins = Math.floor(segundos / 60);
    //nos queda el residuo o sea los segundos restantes
    const secs = segundos % 60;
    // el retorno del tiempo pero en min:seg
    return `${mins.toString().padStart(2, '0')}:${secs.toString().padStart(2, '0')}`;
}
```

En la siguiente parte que es el actualizarTemporizador empezamos con una validación en la que si el juego esta en pausa o el juego termino ya no se podrá actualizar el tiempo lo que significa que se para y lo interrumpimos con el return; siguiendo con lo demás tenemos el restador de segundos con el tiempoRestante--; el que terminaremos seteando en el tiempoDisplay que es nuestro elemento de tiempo que se conecta con nuestra vista jsp para que el tiempo se vea que se esta actualizando también cree unas validaciones en las cuales dice que si el tiempo restante es menor o igual a 60 o sea un min el tiempo se pondrá de color rojo si el tiempo restante es de 2 min el color será naranja y si no es ninguna de las dos el tiempo se quedara igual a como estaba color violeta siguiendo con el code tenemos otro if en el cual nos dice que si el tiempo restante es menor o igual a “0” el temporizador se parara o reiniciar por el clearInterval y el setara nulo para mostrar el mensaje de que se agoto el tiempo y terminara el juego llamando al método terminarJuego();

```
function actualizarTemporizador() {  
    //si el jueo esta pausado o si termino == true significa que retornara y ya no nos  
    // quitara tiempo como es en cascada verifica y si no ya no ejecuta nada mas  
    if (juegoPausado || juegoTerminado) {  
        return;  
    }  
  
    //nos va restando segundo uno por uno de los 6 min que declaramos arriba  
    tiempoRestante--;  
    // se iran restando y al tiempo display se van a ir setiando porque le ponemos el tiempo restante de arriba  
    tiempoDisplay.textContent = formatearTiempo(tiempoRestante);  
  
    // Cambiar color cuando queda poco tiempo a los dos min naranja y al min rojo  
    if (tiempoRestante <= 60) {  
        //si qued un min se pone rojo  
        tiempoDisplay.style.animation = 'pulse 1s infinite';  
        tiempoDisplay.style.color = '#e74c3c';  
    } else if (tiempoRestante <= 120) {  
        //si queda dons min se pone naranja  
        tiempoDisplay.style.color = '#f39c12';  
        tiempoDisplay.style.animation = 'none';  
    } else {  
        // de lo contrario se queda morado  
        tiempoDisplay.style.color = 'blueviolet';  
        tiempoDisplay.style.animation = 'none';  
    }  
  
    //tenemos la aprte si que tiempo restante de 6 min es menor o igual a 0 pues perdio  
    if (tiempoRestante <= 0) {  
        //se reinicia el intervalo de tiempo que es el temporizador entontes queda nullo otra vez y se pausa sol  
        clearInterval(temporizador);  
        temporizador = null;  
        // mostramos el mensaje de que perdio porque se le agoto el tiempo  
        mostrarMensaje('Se te agoto el tiempo por cierto la palabra era: ' + palabraSecreta, 'error');  
        terminarJuego();  
    }  
}
```


Siguiendo tenemos el iniciar juego donde empezamos con un if que dice si el juego esta pausado solamente lo reanudaremos también tenemos otras condiciones donde si el juego ya está iniciado y si no esta terminado no hace nada porque ya se inició, también tenemos la declaración de los estados del juego, ponemos los estados del juego en empezado no pausado y no terminado como ya inicio el juego el botón de iniciar lo hacemos que no se pueda usar. Ahora con las letras por adivinar creamos un array para poder ir almacenando las letras que ya usamos del abecedario, implementamos los intentos de nuevo y los seteamos en el jsp de intentos.

Para seleccionar la palabra utilice el `math.random()` que este imprime de 0-1 pero como lo junto con la longitud de mi array palabra se expande a 5 y esto esta controlado por un `math.floor` para solo traer enteros luego creamos una constante llamada `palabraObj` en la cual implementamos la llamada de un objeto de la lista palabra el cual fue sacado del `math.random`, después tenemos la declaración de `palabraSecreta` por `palabraObj.palabra` que es lo que estaba almacenado en el array palabras y así con el pista e imagen luego para cargar la imagen de la pista mando a traer el `pistaElemento` el cual le seteo el valor de pista que ya fue sacado por el `palabraObj` anteriormente lo mismo para imagen por medio del elemento y el `.src` traigo la imagen guardada actualizo el `displayPalabra` que es donde va la palabra a adivinar, reinicio el dibujo, limpio los comentarios, habilito el teclado inserto mi tiempo a usar, limpiamos el temporizador por si se quedó algún registro guardado y por ultimo mandaremos a actualizar el tiempo cada seg

```
// Iniciar juego
function iniciarJuego() {
  // Si el juego está pausado solamente lo reanudamos y el return solo esta para cortar la ejecución
  if (juegoPausado) {
    reanudarJuego();
    return;
  }

  // juegoIniciado = true y juegoTerminado = false
  // Si el juego ya está iniciado y no terminado no hace nada porque no tiene que volver a reiniciar
  if (juegoIniciado && !juegoTerminado) {
    return;
  }

  // ponemos los estados del juego en empezado no pausado y no terminado
  juegoIniciado = true;
  juegoTerminado = false;
  juegoPausado = false;

  // como ya iniciamos el juego el botón iniciar se cancela
  btnIniciar.disabled = true;
  // le seteamos el valor de iniciado de igual forma ya esta oculto
  btnIniciar.textContent = 'Iniciado';

  // va limpiando nuestras letras del abecedario las que ya fueron usadas
  letrasAdivinadas = [];
  // implementamos los intentos otra vez porque abajo los seteamos
  intentos = 7;
  // seteamos los intentos en el intentosElementos para que se muestren en el jsp
  intentosElementos.textContent = intentos;

  // Seleccionar palabra aleatoria
  // declaramos nuestra constante decimos que esto va a traer numeros enteros y usamos el math.random
  // porque es de 0 a 1 que hace lo random pero si lo multiplicamos por la longitud de nuestro array cabia
  // ya no seria de 01 si no de 0 a lo que este dentro en nuestro caso 5 palabras con su ayuda e imagen
  const indice = Math.floor(Math.random() * palabras.length);
  // en palabraObj seteamos las palabras que van a ser random por medio del indice
  const palabraObj = palabras[indice];

  // aqui ya seteamos los elementos que teniamos antes y a la palabra secreta le damos el valor de palabra de
  // nuestro arraylist y lo mismo para las otras les seteamos los valores que ya tenemos
  palabraSecreta = palabraObj.palabra;
  pista = palabraObj.pista;
  imagenPalabra = palabraObj.imagen;
```



KINAL

Centro Educativo
Técnico Laboral

```
//cargamos la pista en nuestro espacio de jsp con el ${} para dejar nuestra variable llamada
pistaElemento.textContent = `Pista: ${pista}`;

// Mostrar imagen
//en nuestro imagenElement usamos su atributo src para poner la direccion que va a ser igual a la que esta
//guardada en nuestro array donde estan las imagenes ya quemadas
imagenElement.src = imagenPalabra;
//solo es para asegurarnos que la imagen no se quede vacia
imagenElement.style.display = 'block';

// Actualizar ls pantalla si adivino una letra
//actualiza el lugar donde estan las palabras a adivinar
actualizarDisplayPalabra();
//reinicia el dibujo del ahorcado o sea cuando inicia se limpia
resetImagenAhorcado();
// no dejamos ningun estilo a los mensajes
messageElement.style.display = 'none';
//limpamos el mensaje de la partida anterior para que se setee nuevo contenido
messageElement.textContent = '';

// llamamos el metodo para que el teclado sea funcional
habilitarTeclado();

// Iniciar el temporizador de 6 minutos de 60 seg
tiempoRestante = 6 * 60;
//Seteamos el tiempo
tiempoDisplay.textContent = formatearTiempo(tiempoRestante);
tiempoDisplay.style.color = 'blueviolet';
tiempoDisplay.style.animation = 'none';

//volvemos a limpiar el temporizador por si tiene algun dato todavia guardado
if (temporizador) {
    //clear interval limpia todo lo de tiempo o detiene la repeticion del temporizador de 6min
    clearInterval(temporizador);
}

// al iniciar el juego se comienza el conteo del temporizador y se manda a acutalizar para
//que elimine lo segundos cada 1000 o sea cada segundo pe
temporizador = setInterval(actualizarTemporizador, 1000);
}
```

Continuamos con el Reanudar juego en tes solamente tenemos los estados del juego otra vez iguales a como los teníamos en el inicio también deshabilitamos el botón de inicio y activamos nuevamente el teclado, detenemos el tiempo para que no se acumulen intervalos de tiempo y volvemos a activar el temporizador eliminando segundos cada segundo.

```
// Reanudar juego pausado
function reanudarJuego() {
    //todo se encuentra otra vez para ser ejecutado
    juegoPausado = false;
    btnIniciar.disabled = true;
    //volvemos a desaparecer el iniciado
    btnIniciar.textContent = 'Iniciado';
    //habilitamos el teclado
    habilitarTeclado();

    // detenemos el tiempo para que no se acumulen mas intervalos de tiempo
    if (temporizador) {
        clearInterval(temporizador);
    }
    //volvemos a poner el intervalo con la actualizacion de tiempo cada seg
    temporizador = setInterval(actualizarTemporizador, 1000);

    messageElement.style.display = 'none';
}
```

Siguiendo con el code tenemos el pausar juego en el cual verificamos que si el juego ya empezó o si ya termino no hay que pausar nada porque no hay tiempo de ejecución, luego alteramos para que se realice la pausa si el juego pausado era falso se pasa a true para que se apuse y si estaba en true se para a false para reanudar, después tenemos un if el cual es para ver si está en pausa el temporizador se queda quieto y así no nos resta más tiempo, también mostramos un mensaje para decir que está en pausa quitamos el teclado y el botón de inicio ahora se llama reanudar y si no pues el juego sigue.

```
// Pausar juego
function pausarJuego() {
    //si el juego no ha empezado o ya termino no se pausa porque no hay tiempo de ejecucion
    if (!juegoIniciado || juegoTerminado) {
        return;
    }
    //alteramos para que se realice la pausra si el juego pausado era falso se pasa a true para que se ap
    //reanudar
    juegoPausado = !juegoPausado;

    //si el juego se quedo pausado entonces el temporizador se pausara con el clearInterval y elimina las
    if (juegoPausado) {
        if (temporizador) {
            clearInterval(temporizador);
            temporizador = null;
        }

        mostrarMensaje('Haz pusado el juego si quiere seguir jugando haz clic en "Reanudar" para continua
        //como esta pausado el teclado no lo podremos usar entonces mandamos a llamar a su metodo
        deshabilitarTeclado();
        //hacemos que el boton iniciar aparesca otra vez solo que con el nombre de Reanudar
        btnIniciar.disabled = false;
        btnIniciar.textContent = 'Reanudar';
    } else {
        //y si no el juego sigue
        reanudarJuego();
    }
}
```

Creación del teclado esta parte me gusta porque por esto tube que eliminar la lista de botones de teclado que tenía en el jsp entonces aquí empezamos seteando el contenedor Teclado vacío y luego implementamos una constante de letra que tiene de la A-Z luego de esto creamos nuestro for el cual esta para leer las letra de letras lo que itera cada letra de la A-Z después de esto vamos creando los botones de cada letra asignada por el ciclo for le asignamos su css y también seteamos los valores de letra en letra aparte cada botón va a tener su interacción por medio del click para seleccionar la letra que le toque y al final agregamos los botones que vamos creando llegando hasta la Z que es el último valor de letras.

```
// Crear teclado
function crearTeclado() {
  //el contenedor le seteamos un valor vacio para limpiarlo
  contenedorTeclado.innerHTML = '';
  //colocamos las letras que tendra el teclado por default
  const letras = 'ABCDEFGHGIJKLMNOPQRSTUVWXYZ';
  // creamos nuestro buque para leer las palabras ya que este itera capa letra de la A - Z o sea letra por
  for (let letra of letras) {
    const button = document.createElement('button');
    //muestra nuestra letra dentro del boton
    button.textContent = letra;
    //agregamos nuestro css a nuestra letra a cada boton para que se creen con el css definifo
    button.classList.add('btn-letra');
    //a cada boton le vamos a setear los valores de las letras en lentras
    button.dataset.letra = letra;
    // cada boton tiene su funcion por medio del click que seria seleccionar la letra dependiendo del
    button.addEventListener('click', () => manejarClickLetra(letra, button));
    //al final teminamos agregando cada boton al contenedor teclado que es donde estaba mi lista de b
    contenedorTeclado.appendChild(button);
  }
}
```

El manejo de los clicks de cada letra primero decimos que si la letra ya fue adivinada o si el juego no esta iniciado no se van a poder usar también si el juego ya termino o si el juego esta pausado las letras no las podremos usar por eso el return; para que detenga la ejecución de cada una de ellas, siguiendo tenemos el letrasAdivinadas el array creado anteriormente vamos a ir agregando la letra que vamos adivinando dentro luego se ser guardada le decimos que como ya la usamos la tenemos que marcar por medio del css .used y deshabilitamos el botón para que no lo presionen otra vez, después de esto tenemos un if el cual dice que si la palabra secreta que es la que buscamos incluye o tiene un letra que es la que nosotros estamos haciendo click no va a actualizar el display de la palabra para que se muestre y no verificara en caso de que sea la última palabra si no lo es solo sigue verificando hasta que lo sea si no es así los intentos irán disminuyendo y los iremos seteando en el intentosElementos que tiene el id del jsp también va ir mostrando la imagen del ahorcado porque perdió un intento luego tenemos nuestro ultimo if el cual nos dice que si los intentos son menores o iguales a “0” perderemos y nos lanzara un mensaje que perdimos y terminara el juego con el terminarJuego();

```
// Manejar clic en letra
function manejarClickLetra(letra, button) {
    //si la letra ya fue adivinada esa letra ya no se incluye si el juego no esta iniciado no se puede
    //si el juego ya termino ya no se aceptan mas letras y si el juego esta pausado no se usan las letras
    //el return solo detiene la ejecucion si se cumple una de las condiciones
    if (letrasAdivinadas.includes(letra) || !juegoIniciado || juegoTerminado || juegoPausado) {
        return;
    }

    //las letras adivinadas se van guardando en un array letrasAdivinadas y se van agregando al ultimo
    letrasAdivinadas.push(letra);
    //si la letra ya fue usada trae el css de usada y la pone celeste opaco
    button.classList.add('used');
    // y decimos que el boton ya no se puede usar
    button.disabled = true;
    //si la palabraSecreta si incluye la letra
    if (palabraSecreta.includes(letra)) {
        //el display se actualizara e imprimira la letra
        actualizarDisplayPalabra();
        //revisa si todas las letras fueron acertadas si no hasta el otro acierto
        verificarVictoria();
    } else {
        //si la letra no es acertada se restaran los intentos uno por uno
        intentos--;
        //mostraremos los intentos restantes
        intentosElementos.textContent = intentos;
        //mostraremos las partes del ahorcado
        mostrarImagenAhorcado();
        //tambien si los intentos llegan a 0 perderemos y nos mandara un mensaje
        if (intentos <= 0) {
            mostrarMensaje('¡Tan difícil estaba mi bro? La palabra era: ' + palabraSecreta, 'error');
            terminarJuego();
        }
    }
}
```

Actualizar Display palabra en este convertimos la palabra en un array por medio del .split luego del array creamos un map para crear el array ya transformado aplicando la función para ver si la letra está incluida en la letrasAdivinadas y si la letra ya fue encontrada se muestra la letra en el display si no se sigue mostrando el guion casi terminando el array lo pasamos de nuevo a String para insertar espacios entre las letras luego seteamos el valor de display en el espacio de la palabra para mostrarlo en el jsp.

```
// Actualizar el display de la palabra
function actualizarDisplayPalabra() {
    //convierte la palabra en un array o sea Abecedario pasa a A B E C E D A R I O por medio del split('')
    const display = palabraSecreta.split('')
    //map crea el nuevo array transformado aplicando la funcion de cada letra para ver si esta incluida
    //y dice si la letra ya fue adicionada se muestra la letra en el display si no se sigue mostrando
    // si solo encuentra I,O entonces es _ _ _ _ _ I O porque son las unicas que ha encontrado
    .map(letra => letrasAdivinadas.includes(letra) ? letra : '_')
    //convierte el array en string e implementamos la separacion
    .join(' ');
    //seteamos el valor de display en el espacio de la palabra para mostrarlo en el jsp
    espacioPalabra.textContent = display;
}
```

Mostrar imagen del ahorcado. Para esta parte, solo implementamos los intentos. Luego, tenemos un if, el cual, si es mayor a 0 y si los errores son menores o iguales al número de imágenes del array, entonces la imagen se establecerá buscando la imagen 0, luego la 1, y así sucesivamente, hasta que se acaben los intentos . Para resetear, solo dejamos el elemento sin nada y así se limpiará .

```
// Nueva función para mostrar imagen del ahorcado
function mostrarImagenAhorcado() {
    //total de intentos permitidos
    const erroresActuales = 7 - intentos;
    //si el error es mayor a 0 y si los errores son menores o iguales al numero de imagenes del array
    if (erroresActuales > 0 && erroresActuales <= imagenesAhorcado.length) {
        //imagenAhorcado que es el elemento le ingreso una de las imagenes que corresponde al error el m
        imagenAhorcado.src = imagenesAhorcado[erroresActuales - 1];
        imagenAhorcado.style.display = 'block';
    }
}

// Reset de imagen del ahorcado
function resetImagenAhorcado() {
    imagenAhorcado.style.display = 'none';
    //limpia el elemento de la imagen
    imagenAhorcado.src = '';
}
```

Ya casi terminado tenemos el verificar victoria en este solo decimos que si los espacios de la palabra no incluyen _ es porque completo la palabra y gana.

```
// Verificar si gana
function verificarVictoria() {
    //nuestro if en el cual si el espacio de la palabra no incluye ningun _ significa que completo la
    if (!espacioPalabra.textContent.includes('_')) {
        mostrarMensaje('Haz completado la palabra salvaste una vida en este mundo esoooo', 'success');
        terminarJuego();
    }
}
```

Y para terminar la función de terminar juego en este tenemos que el temporizador se pare luego deshabilitamos el teclado y cambiamos los estados del juego:

juego iniciado es falso lo que dice que ya no está activo juego terminado como verdadero lo que dice que ya terminé y el juego pausado en falso porque si está pausado y terminé de todos modos el botón iniciar lo desaparecemos y ya.

```
// Terminar juego
function terminarJuego() {
    //si el temporizador tiene algun intervalo de tiempo lo detiene con el clear y
    //si ya terminé el juego
    if (temporizador) {
        clearInterval(temporizador);
        temporizador = null;
    }
    // ya que terminamos el juego quitamos el teclado porque ya no lo vamos a usar
    deshabilitarTeclado();
    //el juego terminó entonces no está activo
    juegoIniciado = false;
    // marca que la partida ya terminó
    juegoTerminado = true;
    //y si estaba en pausa pues ya terminé
    juegoPausado = false;
    //el botón iniciar aparece otra vez por si queremos volver a jugar
    btnIniciar.disabled = true;
    btnIniciar.textContent = 'Iniciar';
}
```


Explicación extra de cómo se agregaron las palabras y pistas con Base de Datos

Para empezar con esto nosotros debemos tener una base de datos en la cual este creada nuestra entidad Palabra la cual tiene como atributos a la palabra y a la pista que vamos a agregar por otra parte también debemos tener un sp creado previamente para utilizarlo en el PalabrasDAO.

```
create table Palabras(  
    codigoPalabra int auto_increment,  
    palabra varchar(100) not null,  
    pista varchar(200) not null,  
    primary key PK_codigoPalabra(codigoPalabra)  
);
```

Luego de tener esto tenemos que tener nuestra clase Palabras y PalabrasDAO.

En Palabras solamente tenemos que mapear nuestra entidad.

```
Source History  
1 package modelo;  
2  
3 public class Palabras {  
4     private int codigoPalabra;  
5     private String palabra;  
6     private String pista;  
7  
8  
9     public Palabras() {  
10    }  
11  
12     public Palabras(int codigoPalabra, String palabra, String pista) {  
13         this.codigoPalabra = codigoPalabra;  
14         this.palabra = palabra;  
15         this.pista = pista;  
16    }  
17  
18     public int getCodigoPalabra() {  
19         return codigoPalabra;  
20    }  
21  
22     public void setCodigoPalabra(int codigoPalabra) {  
23         this.codigoPalabra = codigoPalabra;  
24    }  
25  
26     public String getPalabra() {  
27         return palabra;  
28    }  
29  
30     public void setPalabra(String palabra) {  
31         this.palabra = palabra;  
32    }  
33  
34     public String getPista() {  
35         return pista;  
36    }  
37  
38     public void setPista(String pista) {  
39         this.pista = pista;  
40    }  
41
```

En Palabras DAO tenemos que crear nuestro método llamado obtenerTodasLasPalabras en el cual nosotros mandamos a llamar nuestro procedimiento almacenado ya creado el cual es el listar luego creamos un while para que recorra nuestra lista de palabras y pueda mostrar el código, palabra y pista que tenemos almacenada en la base de datos luego esto lo almacena en la arraylist llamada palabras y lo retorna para que se vea.

```
package modelo;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class PalabrasDAO {

    private Connection connection;

    public PalabrasDAO(Connection connection) {
        this.connection = connection;
    }

    // Obtener todas las palabras
    public List<Palabras> obtenerTodasLasPalabras() throws SQLException {
        List<Palabras> palabras = new ArrayList<>();
        String sql = "call sp_ListarPalabra()";

        try (PreparedStatement stmt = connection.prepareStatement(sql);
            ResultSet rs = stmt.executeQuery()) {

            while (rs.next()) {
                Palabras palabra = new Palabras(
                    rs.getInt(columnLabel:"codigoPalabra"),
                    rs.getString(columnLabel:"palabra"),
                    rs.getString(columnLabel:"pista")
                );
                palabras.add(palabra);
            }

        }

        return palabras;
    }
}
```

Luego de crear nuestro DAO y nuestro modelo tenemos que crear un nuevo servlet lo odiamos trabajar en el que ya teníamos pero mejor creo otro para que no se vea todo tan junto y pues porque si entonces después de tener esto y crear nuestro servlet dentro de él tenemos que en el método doGet tenemos que poner las peticiones HTTP por medio de la `action=request.getParameter("action")` luego de esto creamos nuestro if para poder decir que si el parámetro action es obtener palabra llame al método correspondiente y si no que se dirige a la página jsp de ahorcado.

Luego tenemos nuestro método llamado obtenerPalabra en este nosotros tenemos que implementar nuestra conexión con la base de datos declarando nuestra variables la cual una de ellas es de ProductosDAO en la cual creamos y hacemos que almacene la conexión del DAO donde ya están las palabras listadas después creamos una lista de Palabras que se llama palabra y le almacenamos el método del dao llamado obtenerTodasLasPalabras para poder listarlos luego hacemos la construcción de nuestro Json para poder iniciar un array Json y como todos los array Json empiezan con [y terminan con] dentro de nuestro array iteramos por medio de un for cada palabra para poder convertir los objetos dentro del array Json con los atributos de codigoPalabra, palabra y pista que es lo que queremos mostrar después de cada uno agregamos las coms para tener espacios menos el ultimo y al final cerramos el array Json con].

Luego de esto hacemos un envío de resultados por medio del establecimiento del tipo de contenido Json y la codificación UTF-8 que hace el UTF-8 simplemente hace que como la letra ñ o una palabra con ó se pueda agregar sin que salga un rombo de que no se encuentra la letra y el `response.getWriter()` tiene in `PrintWriter` para escribir texto en la respuesta HTTP luego él `.write(json.toString())` solo convierte el `StringBuilder` a un tipo string el cual el Json se lo envía al cliente y para terminar con el servlet tenemos una captura de error de tipo SQL la cual si ocurre un error me regresa un error con su mensaje en tipo Json.

Profile Team Tools Window Help Ahorcado_IN5BM - Apache NetBeans IDE 20

499.9/976.0MB Resume (pending events: 16)

asDAO.java × principal.jsp [-/A] × menuRompecabezas.jsp × Validar.java × AhorcadoController.java ×

History

```
Conexion conn = new Conexion();
Connection connection = conn.Conexion();
PalabrasDAO dao = new PalabrasDAO(connection);

List<Palabras> palabras = dao.obtenerTodasLasPalabras();

// Construir JSON manualmente
StringBuilder json = new StringBuilder();
json.append("{");

for (int i = 0; i < palabras.size(); i++) {
    Palabras palabra = palabras.get(index: i);
    json.append("{");
    json.append("\"codigoPalabra\":").append(i: palabra.getCodigoPalabra()).append("\",");
    json.append("\"palabra\":").append(palabra.getPalabra()).append("\",");
    json.append("\"pista\":").append(palabra.getPista()).append("\",");
    json.append("}");

    if (i < palabras.size() - 1) {
        json.append(",");
    }

    json.append("]");

    response.setContentType(type: "application/json");
    response.setCharacterEncoding(charset: "UTF-8");
    response.getWriter().write(s: json.toString());

} catch (SQLException e) {
    response.setStatus(sc: HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
    response.getWriter().write(s: "{\"error\": \"Error al obtener las palabras\"}");
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    doGet(request, response);
}

@Override
public String getServletInfo() {
    return "Servlet para el juego del ahorcado";
}
```

Luego de tener eso implementado solo queda la modificación del js en que partes lo modificamos, la modificación que realice fue referente al arreglo que teníamos de las palabras, pistas e imágenes quemadas ya que lo quitamos y ahora solo las imágenes están quemadas. Si se pregunta el “palabra”: es para indicar que imagen corresponde a que palabra no es que la palabra esta quemada para mostrarla.

```
// Cambiar la declaración del array de palabras
let palabras = [];

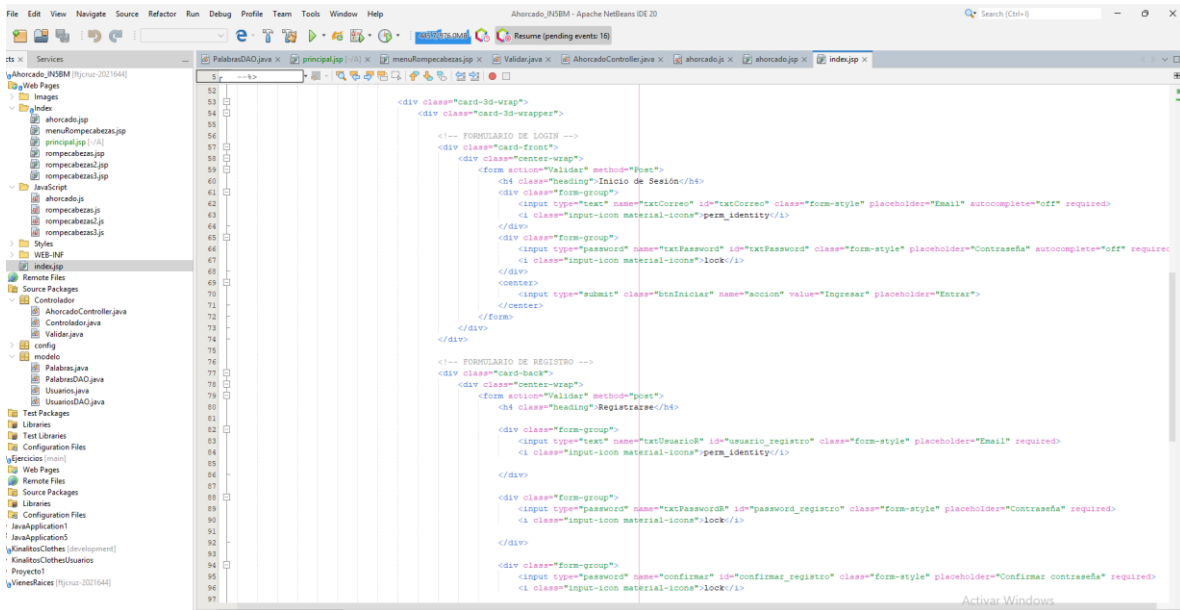
] const imagenesQuemadas = {
  "TORREFACTO": contextPath + "/Images/terrefacto.jpg",
  "SEPTIEMBRE": contextPath + "/Images/septiembre.jpg",
  "MANZANILLA": contextPath + "/Images/manzanilla.jpg",
  "BOMBARDERO": contextPath + "/Images/bombardero.jpg",
  "ABECEDARIO": contextPath + "/Images/abecedario.jpg",
  "PRECIDENTE": contextPath + "/Images/presidente.jpg"
};
```

Luego de esto creamos una nueva función que se llama cargarPalabras() en la cual nosotros retornamos la petición HTTP por medio del fetch() y usamos el contextpath para usar el url que nos lleve a obtener ahorcado controles y haga la acción de obtenerpalabras para que las liste en el DAO, luego tenemos el .then() que nos dice que si la expuesta es falsa entonces que nos muestre el error que la respuesta del servidor dio error y nos convertirá esa respuesta en formato Json en el otro .then solo procesamos los datos decimos que la longitud de las palabras sea luego usamos un for each para iterar sobre la última agregación a la lista de palabras por medio del push() y después retornamos la palabra que se va a alojar en el array palabras=[] entonces en si nosotros pedimos por medio del fetch al servidor el servidor consulta en la base de datos y regresa un Json después el valida su respuesta, pasa de Json a un objeto Js se transforma y se almacena en el array creado de palabras y ya se puede usar la palabra en el juego.

```
function cargarPalabras() {
  return fetch(contextPath + '/AhorcadoController?action=obtenerPalabras')
    .then(response => {
      if (!response.ok) {
        throw new Error('Error en la respuesta del servidor');
      }
      return response.json();
    })
    .then(data => {
      palabras.length = 0;
      data.forEach(item => {
        palabras.push({
          palabra: item.palabra.toUpperCase(),
          pista: item.pista,
          imagen: imagenesQuemadas[item.palabra.toUpperCase()] || contextPath + "/Images/default.jpg"
        });
      });
      return palabras;
    });
}
```

Explicación referente a como se realizó el login y el registrar.

Primero agregamos nuestro jsp con el formulario del login y del registrar.



```

53 </div>
54 <div class="card-3d-wrap">
55 <div class="card-3d-wrapper">
56 <!-- FORMULARIO DE LOGIN -->
57 <div class="card-front">
58 <div class="center-wrap">
59 <form action="Validar" method="post">
60 <h4 class="heading">Inicio de Sesión</h4>
61 <div class="form-group">
62 <input type="text" name="txtCorreo" id="txtCorreo" class="form-style" placeholder="Email" autocomplete="off" required>
63 <i class="input-icon material-icons" perm_identity</i>
64 </div>
65 <div class="form-group">
66 <input type="password" name="txtPassword" id="txtPassword" class="form-style" placeholder="Contraseña" autocomplete="off" required>
67 <i class="input-icon material-icons" lock</i>
68 </div>
69 <div class="center">
70 <input type="submit" class="btnIniciar" name="action" value="Ingresar" placeholder="Entrar">
71 </div>
72 </form>
73 </div>
74 </div>
75 <!-- FORMULARIO DE REGISTRO -->
76 <div class="card-back">
77 <div class="center-wrap">
78 <form action="Validar" method="post">
79 <h4 class="heading">Registrar</h4>
80 <div class="form-group">
81 <input type="text" name="txtUsuarioR" id="usuario_registro" class="form-style" placeholder="Email" required>
82 <i class="input-icon material-icons" perm_identity</i>
83 </div>
84 <div class="form-group">
85 <input type="password" name="txtPasswordR" id="password_registro" class="form-style" placeholder="Contraseña" required>
86 <i class="input-icon material-icons" lock</i>
87 </div>
88 <div class="form-group">
89 <input type="password" name="txtPasswordR" id="password_registro" class="form-style" placeholder="Contraseña" required>
90 <i class="input-icon material-icons" lock</i>
91 </div>
92 <div class="form-group">
93 <input type="password" name="confirmar" id="confirmar_registro" class="form-style" placeholder="Confirmar contraseña" required>
94 <i class="input-icon material-icons" lock</i>
95 </div>
96 </div>
97 </form>
98 </div>
99 </div>

```

Luego de esto tenemos que crear nuestro modelo Usuario en el cual alojamos los atributos de su entidad que esta en la base de datos tal como la contraseña y el correo.

```

package modelo;

public class Usuarios {
    private int codigoUsuario;
    private String contraseña;
    private String correoUsuario;

    public Usuarios() {
    }

    public Usuarios(int codigoUsuario, String contraseña, String correoUsuario) {
        this.codigoUsuario = codigoUsuario;
        this.contraseña = contraseña;
        this.correoUsuario = correoUsuario;
    }

    public int getCodigoUsuario() {
        return codigoUsuario;
    }

    public void setCodigoUsuario(int codigoUsuario) {
        this.codigoUsuario = codigoUsuario;
    }

    public String getContraseña() {
        return contraseña;
    }

    public void setContraseña(String contraseña) {
        this.contraseña = contraseña;
    }

    public String getCorreoUsuario() {
        return correoUsuario;
    }

    public void setCorreoUsuario(String correoUsuario) {
        this.correoUsuario = correoUsuario;
    }
}

```

Luego creamos nuestro UsuariosDAO el cual almacena nuestro agregarUsuario y buscarUsuario.

```
Conexion cn = new Conexion();
Connection con;
PreparedStatement ps;
ResultSet rs;

public Usuarios validar(String correoUsuario, String contraseñaUsuario) {
    Usuarios usuarios = new Usuarios();

    String sql = "call sp_BuscarUsuarioLog(?, ?);";
    try {
        con = cn.Conexion();
        ps = con.prepareStatement(sql);
        ps.setString(parameterIndex: 1, x: correoUsuario);
        ps.setString(parameterIndex: 2, x: contraseñaUsuario);
        rs = ps.executeQuery();

        while (rs.next()) {
            usuarios.setCodigoUsuario(codigoUsuario: rs.getInt(columnLabel: "codigoUsuario"));
            usuarios.setContraseña(contraseña: rs.getString(columnLabel: "contraseña"));
            usuarios.setCorreoUsuario(correoUsuario: rs.getString(columnLabel: "correoUsuario"));
        }
    } catch (Exception e) {
        System.out.println("El usuario o contraseña son incorrectos");
        e.printStackTrace();
    }

    return usuarios;
}

public boolean registrar(String correo, String contraseña) {
    String sql = "call sp_AgregarUsuario(?, ?);";

    try {
        con = cn.Conexion();
        ps = con.prepareStatement(sql);
        ps.setString(parameterIndex: 1, x: correo);
        ps.setString(parameterIndex: 2, x: contraseña);
        int filas = ps.executeUpdate();
        return filas > 0;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}
```

Por ultimo tenemos nuestro validar nuestra opción de ingresar lo trabajamos bajo el método de doPost en el cual comenzamos por medio de un if el cual dice que si Ingresar es igual a la acción que es la petición HTTP entonces va a setear el valor del input txtcorreo dentro de la variable email y lo mismo para la variable pass luego a nuestra variable usuarios le asignamos el .validar email, pass de usuarioDAO y decimos que si el correo de usuario está lleno es porque encontró el usuario en la db luego obtenemos la sesión del usuario por el getSession y con el setAttribute guardamos el objeto usuario en la sesión y por el dispatcher nos lleva a la página principal que significa que pudo entrar y si no nos redirige devuelta al login con mensaje de error

Luego tenemos nuestro Registrar el cual solo asignamos los valores en los txt correspondientes y verificamos que las contraseñas coincidan si no coinciden nos dice que hay un error y que las contraseñas estan malas y nos regresa al index también tenemos el intentar registra al usuario llamando al método registrar del DAO y retorna true si se agrosor y false si no.

Y para ir terminando tenemos un manejo de resultados que dice si nos pudimos registrar nos dice un mensaje que ya se registró el user y si no que no se pudo registrar y nos manda otra vez al index y si la acción no es ni registrar ni ingresar nos redirige al

index.

```
    */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String accion = request.getParameter(name: "accion");

        if ("Ingresar".equalsIgnoreCase(anotherString:accion)) {
            String email = request.getParameter(name: "txtCorreo");
            String pass = request.getParameter(name: "txtPassword");

            usuarios = usuariosDAO.validar(correoUsuario:email, contraseñaUsuario:pass);

            if (usuarios.getCorreoUsuario() != null) {
                HttpSession session = request.getSession();
                session.setAttribute(name: "usuario", value: usuarios);
                request.getRequestDispatcher(path: "Controlador?menu=Principal").forward(request, response);
            } else {
                request.setAttribute(name: "errorLogin", o: "Correo o contraseña incorrectos");
                request.getRequestDispatcher(path: "Controlador?menu=Index").forward(request, response);
            }
        }

        else if ("Registrar".equalsIgnoreCase(anotherString:accion)) {
            String emailR = request.getParameter(name: "txtUsuarioR");
            String passR = request.getParameter(name: "txtPasswordR");
            String confirmar = request.getParameter(name: "confirmar");

            if (!passR.equals(anObject: confirmar)) {
                request.setAttribute(name: "errorRegistro", o: "Las contraseñas no coinciden");
                request.getRequestDispatcher(path: "Controlador?menu=Index").forward(request, response);
                return;
            }

            boolean registrado = usuariosDAO.registrar(correo:emailR, contraseña:passR);

            if (registrado) {
                request.setAttribute(name: "mensajeExito", o: "Usuario registrado con éxito, ahora puedes iniciar sesión")
            } else {
                request.setAttribute(name: "errorRegistro", o: "Error al registrar el usuario");
            }
            request.getRequestDispatcher(path: "Controlador?menu=Index").forward(request, response);
        } else {
            request.getRequestDispatcher(path: "Controlador?menu=Index").forward(request, response);
        }
    }
}
```