Reporte del Proyecto de Ahorcado

Nombre: Alan Francisco Lacán Flores
Carné: 2024010
Sección: IN5BM

Link de Trello:
https://trello.com/invite/b/68b5d3c6832e8be542741d96/ATTId0a97d2ac7926ae45d98864bdc7a6e60764D971B/informatica

Explicación paso a paso del JSP del Index:

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Inicio de Sesión</title>
        <link rel="stylesheet" href="Styles/index.css"/>

    </head>
    <body>
        <div class="login-contendor">
            <h2>Bienvenido</h2>
            <p>Ingresa tus datos para poder iniciar sesion</p>
```

```jsp
    </head>
    <body>
        <div class="login-contendor">
            <h2>Bienvenido</h2>
            <p>Ingresa tus datos para poder iniciar sesion</p>

            <form action="Validar" method="POST">
                <input type="tex" name="txtCorreo" placeholder="Correo" required />
                <input type="password" name="txtContrasena" placeholder="Contraseña" required />
                <button type="submit" name="btnIngresar" value="Ingresar">Iniciar Sesión</button>
            </form>

        </div>
    </body>
</html>
```

En el index lo único que hice fue pedir el correo y contraseña para hacer una validación por medio del Servlet Validar el cual explicare ahora.

Explicación paso a paso del Servlet Validar:

```java
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    // Obtener los parámetros del formulario
    String correo = request.getParameter("txtCorreo");
    String contraseña = request.getParameter("txtContrasena");

    // Validación
    String correoValido = "1";
    String contraseñaValida = "1";

    // Si las credenciales son correctas
    if (correo.equals(correoValido) && contraseña.equals(contraseñaValida)) {
        response.sendRedirect("ahorcado.jsp");
    } else {
        response.sendRedirect("index.jsp");
    }

}
```

En el Servlet de validar cree dos variables tipo String las cuales son el correo y contraseña, luego también dos que van a servir para la validación para que el correo fuera igual a 1 y la contraseña también, en un if si el correo y la contraseña son iguales a 1 pues se cambiara de vista, si no se queda en la misma vista.

Explicación paso a paso del JSP del ahorcado:

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Juego de Ahorcado</title>
        <link rel="stylesheet" href="Styles/ahorcado.css"/>

    </head>
    <body>
        <div class="juego-container">
            <div id="info-juego">
                <h1>Juego del Ahorcado</h1>

                <p>Cronómetro: <span id="cronometro">00:00</span></p>

                <div class="botones">
                    <button id="inicio" onclick="comenzar()">Iniciar</button>
                    <button id="btn-pausar" >Pausar</button>
                    <button id="btn-reiniciar" onclick="reiniciar()">Reiniciar</button>
                </div>

                <div id="pistas">
                    <p id="pista1"></p>
                    <p id="pista2"></p>
                    <p id="pista3"></p>
                </div>

                <p id="palabra-oculta"></p>

                <p>Letras incorrectas: <span id="letras-incorrectas"></span></p>
                <p id="intentos">Intentos restantes: 6</p>

                <input type="text" id="letra" maxlength="1" placeholder="Ingresa una letra">
                <button id="btn-verificar" onclick="verificar()">Probar</button>

                <p id="resultado"></p>
            </div>

            <div id="muñeco-container">
                <img id="muñeco" src="img/Ahorcado.png" alt="Muñeco del Ahorcado">
            </div>
        </div>
        <script src="js/ahorcado.js"></script>
    </body>
</html>
```

Lo primero que hice en el jsp fue ponerle el titulo a la ventana, el link del estilo, en el body fui agregando las diferentes clases que iba a usar en el js, el cronometro, los botones de iniciar, pausar, verificar y reiniciar, coloque las 3 pistas de cada palabra que ayudan a adivinarla, también para que se muestren las distintas letras incorrectas, los intentos que le quedan a la persona, el textField en el cual la persona ingresa la letra que desea, por ultimo agregue la imagen que va cambiando a medida que la persona vaya teniendo más errores y mande a llamar al script con la etiqueta <script></script>.

Explicación paso a paso del JS:

```js
let palabraElegida = "";
let espacios = [];
let letrasMalas = [];
let errores = 0;
let intentos = 6;

let tiempoRestante = 300;
let temporizador;
let juegoEnPausa = false;

const pista1 = document.getElementById("pistaUno");
const pista2 = document.getElementById("pistaDos");
const pista3 = document.getElementById("pistaTres");
const palabraOculta = document.getElementById("palabra-oculta");
const letrasIncorrectasEl = document.getElementById("letras-incorrectas");
const intentosEl = document.getElementById("intentos");
const imagen = document.getElementById("muñeco");
const cronometroEl = document.getElementById("cronometro");
const btnPausar = document.getElementById("btn-pausar");
```

Declare las variables como let, las cuales van a ir cambiando mientras la persona va jugando: palabraElegida que es igual “ ”, espacios que es un arreglo al igual que letras malas, errores que empezara en 0 hasta llegar a 6 va a perder la persona, y por ultimo los 6 intentos que tendrá la persona.

Declare las variables con const, lo que hacen estas variables es que están obteniendo al elemento del jsp, eso lo hice con las pistas, la palabra oculta, las letras incorrectas, intentos y el muñeco de la imagen.

```
async function cargarPalabraDesdeBD() {
    try {
        const respuesta = await fetch("ControladorAhorcado?action=obtenerPalabras", {
            headers: { "Accept": "application/json" }
        });
        if (!respuesta.ok) throw new Error("No se pudo obtener palabras del servidor");

        const data = await respuesta.json();
        if (!Array.isArray(data) || data.length === 0) {
            throw new Error("No hay palabras en la base de datos");
        }

        const obj = data[Math.floor(Math.random() * data.length)];
        palabraElegida = (obj.palabra || obj.Palabra || "").toUpperCase();
        if (!palabraElegida) throw new Error("El objeto recibido no contiene la palabra");

        espacios = Array(palabraElegida.length).fill("_");
        palabraOculta.textContent = espacios.join(" ");

        pista1.textContent = "Pista No.1: " + (obj.pistaUno || obj.pista || "");
        pista2.textContent = "Pista No.2: " + (obj.pistaDos || "");
        pista3.textContent = "Pista No.3: " + (obj.pistaTres || "");
    } catch (err) {
        console.error(err);
        alert("⚠ " + err.message);
    }
}
```

En esta función uso el async ya que es el que se comunica con el ControladorAhorcado para que se obtenga la palabra, luego declare con const la respuesta que va a ser la que buscar por medio del fetch en “ControladorAhorcado?action=obtenerPalabras”, si la respuesta no obtiene una palabra se muestra un mensaje de que no se obtuvo la palabra.

La cons obj elige una palabra aleatoriamente y se le asigna a la palabraElegida.

Se crea un array con los espacios necesarios para esa palabra, por último se mandan a llamar las pistas que vienen del servidor para que se muestren y así la persona puede tener pistas de que palabra es.

```
async function comenzar() {
    if (!palabraElegida) {
        await cargarPalabraDesdeBD();

        if (!palabraElegida) {
            // fallback si no hay BD
            const palabraAleatoria = palabrasAhorcado[Math.floor(Math.random() * palabrasAhorcado.length)];
            palabraElegida = palabraAleatoria.palabra.toUpperCase();
            espacios = Array(palabraElegida.length).fill("_");
            palabraOculta.textContent = espacios.join(" ");
            pista1.textContent = "Pista No.1: " + palabraAleatoria.pistas[0];
            pista2.textContent = "Pista No.2: " + palabraAleatoria.pistas[1];
            pista3.textContent = "Pista No.3: " + palabraAleatoria.pistas[2];
        }

        tiempoRestante = 300;
        cronometroEl.textContent = tiempoRestante;
        iniciarCronometro();
    }
}
```

La función comenzar(), esta función empieza al darle clic al botón iniciar, declare un if que lo que hace es poner una palabra aleatoria de las que se declararon en el arreglo, se agregaron los espacios, las pistas y así cuando la persona comienza ya se muestran las diferentes pistas y los espacios para poner cada letra.

El tiempoRestante = 300 segundos que equivalen a 5 minutos.
luego cuando la persona le de al botón iniciar empezara a disminuir el tiempoRestante.

```
function reiniciar() {
    palabraElegida = "";
    espacios = [];
    letrasMalas = [];
    errores = 0;
    intentos = 6;

    palabraOculta.textContent = "";
    letrasIncorrectasEl.textContent = "";
    intentosEl.textContent = "Intentos restantes: 6";
    pista1.textContent = "";
    pista2.textContent = "";
    pista3.textContent = "";
    imagen.src = "img/Ahorcado.png";

    tiempoRestante = 300;
    cronometroEl.textContent = tiempoRestante;
    juegoEnPausa = false;
    btnPausar.textContent = "Pausar";

    comenzar();
}
```

La función reiniciar() lo único que hace es cuando la persona le da en reiniciar se limpia todas las variables al igual que la imagen.

Por ultimo al darle reiniciar se llama al método Comenzar() para que muestre otras pistas y así adivinar otra palabra.

```
function verificar() {
    const verificarLetra = document.getElementById("letra");
    const letra = verificarLetra.value.toUpperCase();
    verificarLetra.value = "";

    if (!letra || letrasMalas.includes(letra) || espacios.includes(letra)) return;

    if (palabraElegida.includes(letra)) {
        for (let i = 0; i < palabraElegida.length; i++) {
            if (palabraElegida[i] === letra) {
                espacios[i] = letra;
            }
        }
        palabraOculta.textContent = espacios.join(" ");
    } else {
        letrasMalas.push(letra);
        letrasIncorrectasEl.textContent = letrasMalas.join(", ");
        errores++;
        intentos--;
        intentosEl.textContent = `Intentos restantes: ${intentos}`;
        imagen.src = `img/Ahorcado${errores}.png`;
    }

    if (!espacios.includes("_")) {
        alert("¡Ganaste!");
    }

    if (errores === 6) {
        alert("¡Perdiste! La palabra era: " + palabraElegida);
    }
}
```

La función verificar(), se crean dos constantes en la cual una obtiene lo que puso la persona en el input, luego la letra se vuelve mayúscula para que coincida con las palabras agregadas en el arreglo.

El primer if lo que hace es que si esa letra ya la puso y esta buena o mala ya no la incluye.

El segundo if lo que hace es recorrer toda la palabra y va viendo si incluye la letra esa palabra, si la letra es incorrecta se agrega en el arreglo de letras incorrectas, se muestran todas las letras incorrectas, aumentan los errores y disminuye los intentos, por ultimo la imagen va subiendo 1, es decir que si era Ahorcado ahora cambiara a Ahorcado1.

El tercer if lo que hace es que si los guiones se llenaron manda una alerta de que la persona ganó.

El cuarto if lo que hace es que si los errores son == a 6 pues manda una alerta de que la persona perdió.

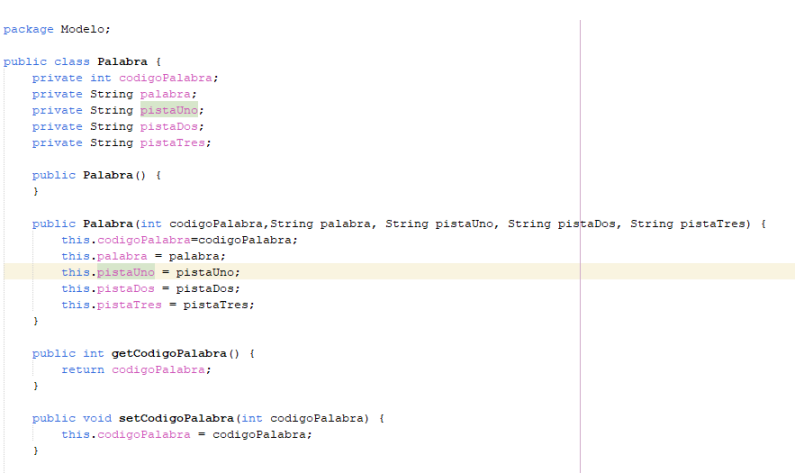
```
function actualizarCronometro() {
    if (!juegoEnPausa && tiempoRestante > 0) {
        tiempoRestante--;
        cronometroEl.textContent = tiempoRestante;
    } else if (tiempoRestante <= 0) {
        clearInterval(temporizador);
        alert("¡Se acabó el tiempo! La palabra era: " + palabraElegida);
        reiniciar();
    }
}

function iniciarCronometro() {
    clearInterval(temporizador);
    temporizador = setInterval(actualizarCronometro, 1000);
}

function pausarJuego() {
    juegoEnPausa = !juegoEnPausa;
    btnPausar.textContent = juegoEnPausa ? "Reanudar" : "Pausar";
}
```

La función actualizarCronometro() si el juego no esta en pausa y es mayo a 0 el tiempo va a estar disminuyendo y si el tiempo es mejor a 0 se manda una alerta que se acabo el tiempo.

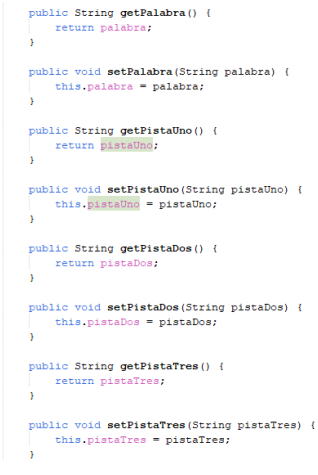
La función iniciarCronometro() limpia el temporizador y el temporizador va a volver a ser 300 segundos.

La función pausarJuego() lo único que hace es que si le dan al botón pausar el cronometro se queda parado.

Explicación paso a paso de la Clase Palabra:



```
package Modelo;

public class Palabra {
    private int codigoPalabra;
    private String palabra;
    private String pistaUno;
    private String pistaDos;
    private String pistaTres;

    public Palabra() {
    }

    public Palabra(int codigoPalabra,String palabra, String pistaUno, String pistaDos, String pistaTres) {
        this.codigoPalabra=codigoPalabra;
        this.palabra = palabra;
        this.pistaUno = pistaUno;
        this.pistaDos = pistaDos;
        this.pistaTres = pistaTres;
    }

    public int getCodigoPalabra() {
        return codigoPalabra;
    }

    public void setCodigoPalabra(int codigoPalabra) {
        this.codigoPalabra = codigoPalabra;
    }
```



```
    public String getPalabra() {
        return palabra;
    }

    public void setPalabra(String palabra) {
        this.palabra = palabra;
    }

    public String getPistaUno() {
        return pistaUno;
    }

    public void setPistaUno(String pistaUno) {
        this.pistaUno = pistaUno;
    }

    public String getPistaDos() {
        return pistaDos;
    }

    public void setPistaDos(String pistaDos) {
        this.pistaDos = pistaDos;
    }

    public String getPistaTres() {
        return pistaTres;
    }

    public void setPistaTres(String pistaTres) {
        this.pistaTres = pistaTres;
    }
}
```

En la clase Palabra lo único que hice fue declarar los atributos de la entidad Palabra, luego el constructor vacío y lleno y por ultimo los getters y setters.

Explicación paso a paso de la Clase PalabraDAO:

```
public List<Palabra> listar() {
    List<Palabra> lista = new ArrayList<>();
    String sql = "SELECT * FROM Palabras";
    try {
        con = cn.Conexion();
        ps = con.prepareStatement(sql);
        rs = ps.executeQuery();

        while (rs.next()) {
            Palabra p = new Palabra();
            p.setCodigoPalabra(rs.getInt("codigoPalabra"));
            p.setPalabra(rs.getString("palabra"));
            p.setPistaUno(rs.getString("pistaUno"));
            p.setPistaDos(rs.getString("pistaDos"));
            p.setPistaTres(rs.getString("pistaTres"));
            lista.add(p);
        }
    } catch (Exception e) {
        System.out.println("Error al listar palabras: " + e);
    }
    return lista;
}
```

En la clase PalabraDAO solo cree el método para listar todas las palabras en el cual se crea una variable tipo String que va servir para llamar el “SELECT * FROM Palabras”, en el try catch se maneja un while en el cual se instancia la clase Palabra y también se esta seteando cada atributo y por último a la lista se están agregrando todos los atributos, en el catch solo se maneja la excepción, por último, solo se retorna la lista.

Explicación paso a paso del Servlet ControladorAhorcado:

```
private void obtenerPalabras(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    // Usar tu clase de conexión
    Conexion conn = new Conexion();
    Connection connection = conn.Conexion();
    PalabraDAO dao = new PalabraDAO(connection);
    List<Palabra> palabras = dao.listar();
    // Construir JSON manualmente
    StringBuilder json = new StringBuilder();
    json.append("[");
    for (int i = 0; i < palabras.size(); i++) {
        Palabra palabra = palabras.get(i);
        json.append("{");
        json.append("\"codigoPalabra\":").append(palabra.getCodigoPalabra()).append(",");
        json.append("\"palabra\":\"").append(palabra.getPalabra()).append("\",");
        json.append("\"pistaUno\":\"").append(palabra.getPistaUno()).append("\",");
        json.append("\"pistaDos\":\"").append(palabra.getPistaDos()).append("\",");
        json.append("\"pistaTres\":\"").append(palabra.getPistaTres()).append("\"");
        json.append("}");

        if (i < palabras.size() - 1) {
            json.append(",");
        }
    }
    json.append("]");
    response.setContentType("application/json");
    response.setCharacterEncoding("UTF-8");
    response.getWriter().write(json.toString());
}
```

En el método obtenerPalabras se crea al objeto Conexión que es la que gestiona la conexión con la base de datos, en el objeto PalabraDAO en el cual se esta usando la conexión, luego el dao.listar() es el que devuelve la lista de las palabras y pistas.

En la construcción del JSON se crea un for en la cual cada objeto del Array representa la palabra con sus campos los cuales son: codigoPalabra, palabra, pistaUno, pistaDos, pistaTres.

El if solo es para poner comas entre los objetos.