



PLAN DE TRABAJO DEL ESTUDIANTE

DATOS DEL ESTUDIANTE

Apellidos y Nombres:	Pachas Saavedra Jesús Omar	ID:	001514827
Dirección Zonal/CFP:	ICA – AYACUCHO / CFP CHINCHA		
Carrera:	Ingeniería de software con inteligencia artificial	Semestre:	V
Curso/ Mód. Formativo:	TALLER DE DESARROLLO DE APLICACIONES CON MACHINE LEARNING		
Tema de Trabajo Final:	Implementación de un sistema automatizado de gestión documental y selección de currículos mediante machine learning en una municipalidad		

1. INFORMACIÓN**▪ Identifica la problemática del caso práctico propuesto.**

La problemática del caso práctico es que la Municipalidad Provincial de Yau presenta inconvenientes en la gestión de trámites administrativos debido a que cuentan con sistema manual que genera largas esperas, errores frecuentes y falta de control sobre los datos procesados. Esto ha afectado tanto a los ciudadanos como al personal administrativo, deteriorando la calidad del servicio y la imagen institucional. La ausencia de automatización y tecnologías como el machine learning impide identificar y resolver cuellos de botella, dificultando la mejora continua del proceso de atención al ciudadano.

▪ Identifica propuesta de solución y evidencias.

La propuesta de solución es desarrollar e implementar un sistema automatizado de gestión de trámites que utilice algoritmos de machine learning para analizar, priorizar y optimizar el flujo de solicitudes en la municipalidad. El sistema debe reducir errores en el procesamiento, identificar patrones de congestión y prever posibles fallos, además de incluir un mecanismo de alertas para mantener informados a los ciudadanos sobre el estado de sus trámites en tiempo real. De esta forma, se mejorará la eficiencia operativa y la satisfacción ciudadana.

▪ **Respuestas a preguntas guía**

Durante el análisis y estudio del caso práctico, debes obtener las respuestas a las interrogantes:

Pregunta 01:	¿Cuáles son los principales trámites que generan más retrasos y frustración entre los ciudadanos en la municipalidad?
Los principales trámites que suelen generar más retrasos y frustración entre los ciudadanos en la municipalidad son los que requieren múltiples validaciones y aprobaciones, como la solicitud de licencias de construcción, permisos de funcionamiento, y trámites relacionados con servicios públicos (agua, luz, saneamiento).	
Pregunta 02:	¿Qué datos históricos son necesarios para entrenar el modelo de machine learning y cómo se pueden recolectar de manera efectiva?
<p>Para entrenar el modelo de machine learning considero los siguientes datos:</p> <ul style="list-style-type: none"> • Tipo de trámite realizado (licencias, permisos, servicios, etc.) • Tiempo de inicio y fin de cada trámite • Registro de errores o incidencias ocurridas durante el proceso <p>Para recolectar estos datos de manera efectiva, se debe digitalizar la información histórica existente, implementar un sistema digital para registrar en tiempo real los trámites, y capacitar al personal para asegurar la correcta y constante entrada de datos, garantizando la calidad y confiabilidad de la información.</p>	
Pregunta 03:	¿Qué métricas se utilizarán para evaluar la efectividad del sistema de machine learning implementado?
Para evaluar la efectividad, se utilizarán métricas clave como la precisión, que mide el porcentaje de trámites correctamente clasificados o priorizados; el recall, que evalúa la capacidad del modelo para identificar adecuadamente los trámites críticos que requieren atención urgente; y el tiempo promedio de respuesta, que permite medir la reducción en la duración del procesamiento de trámites en comparación con el sistema manual anterior.	
Pregunta 04:	¿Cómo se garantizará la aceptación y uso del nuevo sistema por parte del personal de la municipalidad?
Para garantizar la aceptación y uso del nuevo sistema, se implementará un programa de capacitación continua que facilite el aprendizaje y manejo del sistema, acompañado de sesiones de demostración prácticas y materiales didácticos accesibles.	
Pregunta 05:	¿Qué mecanismos de feedback se establecerán para ajustar el sistema en función de la experiencia del ciudadano y del personal?
Se establecerán mecanismos de feedback mediante encuestas periódicas tanto para los ciudadanos como para el personal administrativo, que permitan recoger opiniones sobre la usabilidad, eficiencia y satisfacción del sistema.	

2. PLANIFICACIÓN DEL TRABAJO

▪ **Cronograma de actividades:**

N°	ACTIVIDADES	CRONOGRAMA					

▪ **Lista de recursos necesarios:**

1. MÁQUINAS Y EQUIPOS	
Descripción	Cantidad
Laptop	

2. HERRAMIENTAS E INSTRUMENTOS	
Descripción	Cantidad
Processing	
Visual Studio Code	

3. MATERIALES E INSUMOS	
Descripción	Cantidad
Anotaciones de clases	
Videos de youtube	

3. DECIDIR PROPUESTA

- Describe la propuesta determinada para la solución del caso práctico

PROPUESTA DE SOLUCIÓN

La propuesta de solución para el caso práctico consiste en implementar un sistema automatizado de gestión de trámites con machine learning, que permita optimizar y agilizar la atención al ciudadano en la municipalidad. Este sistema utilizará algoritmos de aprendizaje automático para analizar datos históricos de los trámites, identificar patrones de congestión y priorizar aquellos trámites críticos que requieren atención inmediata. Además, incorporará un sistema de alertas que notifique a los ciudadanos sobre el estado de sus solicitudes en tiempo real, reduciendo así la incertidumbre y mejorando la experiencia del usuario. De esta forma, se busca disminuir el tiempo de espera, minimizar los errores en el procesamiento y mejorar la eficiencia operativa del personal administrativo, mejorando la percepción y calidad del servicio público ofrecido.

4. EJECUTAR

- Resolver el caso práctico, utilizando como referencia el problema propuesto y las preguntas guía proporcionadas para orientar el desarrollo.
- Fundamentar sus propuestas en los conocimientos adquiridos a lo largo del curso, aplicando lo aprendido en las tareas y operaciones descritas en los contenidos curriculares.

INSTRUCCIONES: Ser lo más explícito posible. Los gráficos ayudan a transmitir mejor las ideas. Tomar en cuenta los aspectos de calidad, medio ambiente y SHI.

[illegible]

DIBUJO / ESQUEMA / DIAGRAMA DE PROPUESTA

(Adicionar las páginas que sean necesarias)

Clasificador de triángulos

Antes de comenzar con el desarrollo del proyecto, es necesario contar con un conjunto de imágenes que permitan entrenar nuestro modelo de reconocimiento. Para esto, utilizare processing, un entorno para generar imágenes de seis tipos diferentes de triángulos. Cada imagen lo dibujare en un lienzo pequeño de 64x64 píxeles, con una rotación aleatoria ligera para simular distintas orientaciones. Así, se crean 100 variaciones por cada tipo de triángulo, lo que da un total de 600 imágenes que servirán como datos de entrenamiento.



Archivo pruebas.pde

void setup() :

En esta función establezco el tamaño del lienzo en 64x64 píxeles para mantener las imágenes pequeñas y optimizar el uso de recursos. Finalmente, uso noLoop() para que el programa dibuje una sola vez y no se repita constantemente.

void draw()

Aquí genero las imágenes de los seis tipos de triángulos. Uso un ciclo que recorre los seis triángulos y, para cada tipo, genero 100 imágenes con una ligera rotación aleatoria para variar la orientación. Cada triángulo se dibuja en el centro del lienzo y luego se guarda en la carpeta data con un nombre que identifica su tipo y número.

```

1 void setup() {
2   pixelDensity(1);
3   size(64, 64);
4   noLoop();
5 }
6
7 void draw() {
8   for (int t = 0; t < 6; t++) {
9     String prefix = "";
10    if (t == 0) prefix = "trianguloRTA";
11    if (t == 1) prefix = "trianguloRTB";
12    if (t == 2) prefix = "trianguloRTC";
13    if (t == 3) prefix = "trianguloRTAD";
14    if (t == 4) prefix = "trianguloACUTA";
15    if (t == 5) prefix = "trianguloACUTB";
16
17    for (int i = 1; i <= 100; i++) {
18      background(255);
19      strokeWeight(4);
20      stroke(100, 100, 100);
21      fill(255);
22
23      pushMatrix();
24      translate(width / 2, height / 2);
25      rotate(random(-0.08, 0.08));
26
27      if (t == 0) drawRectATriangle();
28      if (t == 1) drawRectBTriangle();
29      if (t == 2) drawRectDTriangle();
30      if (t == 3) drawRectCTriangle();
31      if (t == 4) drawAcuteATriangle();
32      if (t == 5) drawAcuteBTriangleUpsideDown();
33
34      popMatrix();
35
36      saveTrimmed(prefix, i);
37    }
38  }
39  exit();
40 }

```

void drawRectATriangle()

Dibujó un triángulo rectángulo orientado de forma específica, con vértices posicionados para formar un triángulo rectángulo "A". Utilizo una longitud fija para los lados, basada en la variable s.

void drawRectBTriangle()

Aca dibujó un segundo triángulo rectángulo con una orientación diferente, denominado "B". La forma se consigue con vértices que definen un triángulo rectángulo con sus lados en distintas posiciones respecto al lienzo.

void drawRectCTriangle()

Dibujo un tercer triángulo rectángulo con otra orientación, llamado "C". Se definen los vértices para que forme este triángulo con base y altura orientadas de forma distinta a los anteriores.

void drawRectDTriangle()

Dibujo un cuarto triángulo rectángulo, denominado "D", con una orientación diferente a los anteriores, también basado en la variable *s* para definir el tamaño.

void drawAcuteATriangle()

Dibujo un triángulo acutángulo (con ángulos menores a 90°), llamado "Acute A". Utilizo una escala ligeramente menor para los lados, y defino vértices para crear esta forma específica.

void drawAcuteBTriangleUpsideDown()

Dibujo otro triángulo acutángulo, llamado "Acute B", pero esta vez orientado hacia abajo (invertido). Los vértices están ajustados para crear esta forma particular.

```

42 void drawRectATriangle() {
43     float s = 24;
44     triangle(-s, s, s, s, -s, -s);
45 }
46
47 void drawRectBTriangle() {
48     float s = 24;
49     triangle(s, s, s, -s, -s, s);
50 }
51
52 void drawRectCTriangle() {
53     float s = 24;
54     triangle(s, -s, -s, -s, s, s);
55 }
56
57 void drawRectDTriangle() {
58     float s = 24;
59     triangle(-s, -s, -s, s, s, -s);
60 }
61
62 void drawAcuteATriangle() {
63     float s = 22;
64     triangle(0, -s, s * 0.87, s * 0.5, -s * 0.87, s * 0.5);
65 }
66
67 void drawAcuteBTriangleUpsideDown() {
68     float s = 24;
69     triangle(0, s, -s * 0.95, -s * 0.31, s * 0.6, -s * 0.8);
70 }

```

void saveTrimmed(String prefix, int count)

Aquí guardo la imagen actual del lienzo como un archivo PNG en la carpeta data. El nombre del archivo incluye un prefijo que indica el tipo de triángulo y un número con tres dígitos para identificar la imagen única. Esto permite organizar fácilmente las imágenes generadas.

```
72 void saveTrimmed(String prefix, int count) {  
73     PImage trimmed = get(0, 0, width, height);  
74     String filename = String.format("data/%s%03d.png", prefix, count);  
75     trimmed.save(filename);  
76 }
```

Carpeta Data

Dentro del proyecto, crearé una carpeta llamada data donde se almacenarán todas las imágenes generadas. Esta carpeta es fundamental porque en ella se guardarán los 100 ejemplos de cada uno de los seis tipos de triángulos que estoy generando. Al usar el método saveTrimmed(), las imágenes se guardarán automáticamente en esa ubicación con nombres que indican el tipo de triángulo y su número correspondiente. Esta organización me permitirá tener todos los datos visuales listos para futuras etapas, como el entrenamiento o la clasificación de triángulos.

Archivo entrenamiento.html**Inclusión de librerías :**

Aquí importo las librerías necesarias para poder realizar el entrenamiento:

- p5.js, que me permite trabajar con imágenes y dibujar en el lienzo.
- ml5.js, que me da acceso a herramientas de inteligencia artificial (IA) de forma sencilla, en este caso, para entrenar un clasificador de imágenes.

Función preload() :

En esta función cargo las 100 imágenes de cada tipo de triángulo desde la carpeta data. Uso la función nf() para que el índice tenga siempre tres cifras (por ejemplo, 001, 002, ..., 100), asegurando que coincidan con los nombres generados previamente en Processing.

```

7   <script src="https://cdn.jsdelivr.net/npm/p5@1.4.0/lib/p5.min.js"></script>
8   <script src="https://cdn.jsdelivr.net/npm/ml5@0.12.2/dist/ml5.min.js"></script>
9   </head>
10  <body>
11  <script>
12    let trianguloRectA = [];
13    let trianguloRectB = [];
14    let trianguloRectC = [];
15    let trianguloRectD = [];
16
17    let trianguloAcutA = [];
18    let trianguloAcutB = [];
19
20    let ShapeClassifier;
21
22    function preload() {
23      for (let i = 0; i < 100; i++) {
24        let index = nf((i + 1), 3, 0);
25
26        trianguloRectA[i] = loadImage(`data/trianguloRTA${index}.png`);
27        trianguloRectB[i] = loadImage(`data/trianguloRTB${index}.png`);
28        trianguloRectC[i] = loadImage(`data/trianguloRTC${index}.png`);
29        trianguloRectD[i] = loadImage(`data/trianguloRTAD${index}.png`);
30
31        // Acutángulos tipo A, B
32        trianguloAcutA[i] = loadImage(`data/trianguloACUTA${index}.png`);
33        trianguloAcutB[i] = loadImage(`data/trianguloACUTB${index}.png`);
34      }
35    }

```

Función setup():

Aquí configuro el clasificador con imágenes de 64x64 píxeles y 4 canales (RGBA). Luego le proporciono los datos: 100 imágenes por cada clase de triángulo, con sus respectivas etiquetas. Después normalizo los datos (esto es importante para que el entrenamiento sea correcto) y finalmente inicio el proceso de entrenamiento con 50 épocas.

Función finishedTraining():

Cuando el entrenamiento termina, esta función se ejecuta. En mi caso, solo muestro un mensaje en la consola y luego guardo el modelo entrenado para poder usarlo más adelante sin necesidad de volver a entrenarlo.

```

22  function preload() {
23    for (let i = 0; i < 100; i++) {
24      let index = nf((i + 1), 3, 0);
25
26      trianguloRectA[i] = loadImage(`data/trianguloRTA${index}.png`);
27      trianguloRectB[i] = loadImage(`data/trianguloRTB${index}.png`);
28      trianguloRectC[i] = loadImage(`data/trianguloRTC${index}.png`);
29      trianguloRectD[i] = loadImage(`data/trianguloRTAD${index}.png`);

```


Archivos generados tras el entrenamiento

Una vez que se ejecuta el archivo HTML con el código de entrenamiento, ml5.js generara automáticamente tres archivos:

1. **model.json**

Este archivo contiene la estructura del modelo entrenado (capas, configuración, arquitectura, etc.). Es esencial para poder usar el modelo más adelante para clasificar nuevas imágenes.

2. **model.weights.bin**

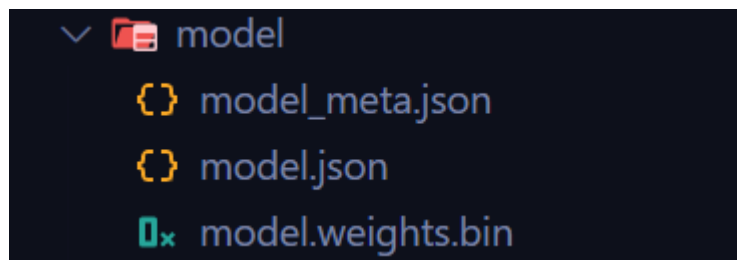
Este archivo binario contiene los pesos entrenados del modelo. Es el resultado de todo el aprendizaje realizado a partir de las imágenes de entrenamiento.

3. **metadata.json**

Incluye información adicional como las etiquetas de las clases utilizadas, el número de datos de entrenamiento por clase, y otros detalles del modelo.

Estos 3 archivos los colocaré dentro de una carpeta llamada model. Esta carpeta estará ubicada dentro del mismo directorio del proyecto.

El objetivo de esto es mantener una mejor organización del proyecto y facilitar el acceso al modelo entrenado cuando desee cargarlo posteriormente para realizar predicciones. Esto también evita confusión con otros archivos del sistema y mantiene el código más limpio y mantenible.



Archivo consume-rn.html

setup()

Primero inicializo el canvas donde el usuario dibujara y establezco un fondo blanco. Configuro la red neuronal de ml5 para clasificación de imágenes con una entrada de 64x64x4. Creo un botón para limpiar el canvas y un contenedor para mostrar resultados.

modelLoaded()

Esta función se ejecuta tras cargar correctamente el modelo. Cambio el estado para indicar que el modelo está listo y actualizo el mensaje en pantalla para informar al usuario que puede comenzar a dibujar.

mouseReleased()

Detecto cuando el usuario suelta el botón del mouse. Si el modelo está cargado y el usuario ha dibujado algo, llamo a la función que procesa y clasifica la imagen dibujada.

```

57     function setup() {
58         canvas = createCanvas(400, 400);
59         background(255);
60
61         let options = {
62             inputs: [64, 64, 4],
63             task: "imageClassification"
64         };
65         ShapeClassifier = ml5.neuralNetwork(options);
66
67         const modelDetails = {
68             model: 'model/model.json',
69             metadata: 'model/model_meta.json',
70             weights: 'model/model.weights.bin'
71         };
72         You, 20 hours ago • reconocimiento de triángulos v-1.0
73         clearButton = createButton('Limpiar');
74         clearButton.mousePressed(() => {
75             background(255);
76             quitarResaltado();
77             resultDiv.html('Dibuja un triángulo para clasificar.');
```

classifyImage()

Capturo la imagen actual del canvas con get(), la redimensiono a 64x64 píxeles para cumplir con la entrada del modelo, y envío esta imagen a la red neuronal para obtener la clasificación.

```
function classifyImage() {
  let img = get();
  img.resize(64, 64);
  ShapeClassifier.classify({ image: img }, gotResults);
}
```

gotResults()

Recibo los resultados de la clasificación. Verifico que no haya errores y que la confianza sea superior al umbral (0.5). Si la etiqueta es distinta a la anterior y la confianza es alta, actualizo la interfaz, hablo el resultado y resalto la casilla correspondiente. Si la confianza es baja o hay error, limpio el resaltado y notifico que no se pudo clasificar.

```
function gotResults(err, results) {
  if (err) {
    console.error(err);
    return;
  }

  if (results && results[0]) {
    let etiqueta = results[0].label.toLowerCase();
    let confianza = results[0].confidence;

    if (confianza > 0.5 && ultimaEtiqueta !== etiqueta && dibujoRealizado) {
      ultimaEtiqueta = etiqueta;

      const tipoMap = {
        rectangulo: "rectángulo",
        acutangulo: "acutángulo"
      };

      let letra = etiqueta.slice(-1).toUpperCase();
      let tipoClave = etiqueta.slice(0, -1);

      let tipo = tipoMap[tipoClave] || tipoClave;

      let textoDecir = `Este es un triángulo ${tipo} tipo ${letra}`;

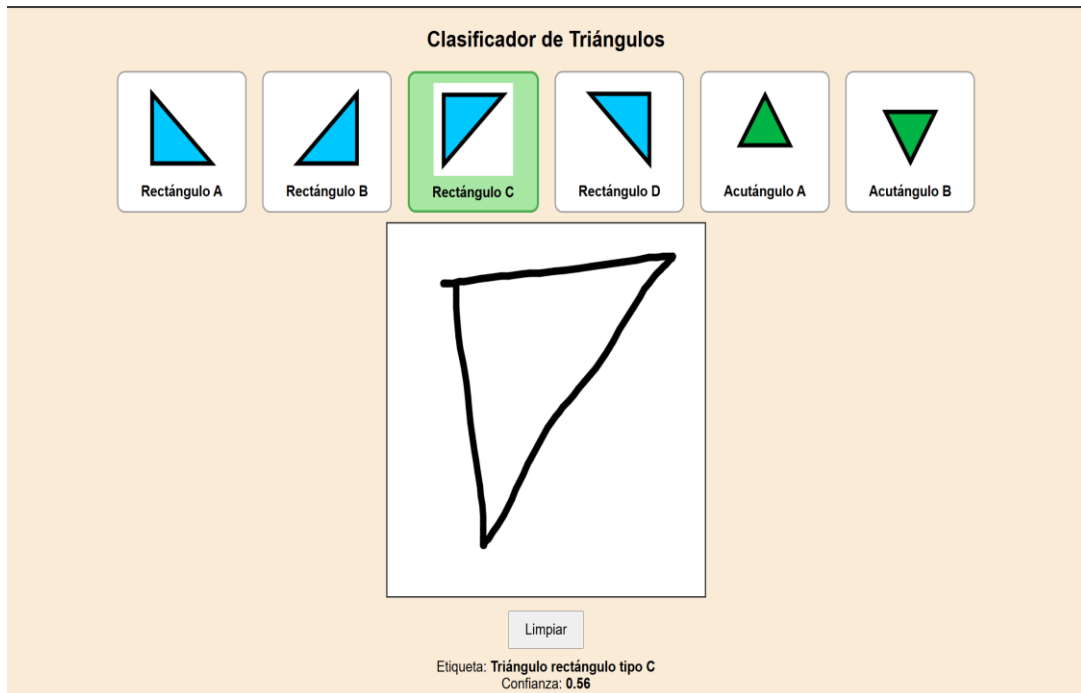
      resultDiv.html(`Etiqueta: <strong>Triángulo ${tipo} tipo ${letra}</strong><br>Confianza: <strong>${nf(confianza, 0, 2)}</strong>`);
      decir(textoDecir);
      resaltarCasilla(etiqueta);
    } else if (confianza <= 0.5) {
      resultDiv.html("No se pudo clasificar con suficiente confianza.");
      quitarResaltado();
      ultimaEtiqueta = "";
    }
  } else {
    resultDiv.html("No se pudo clasificar.");
    quitarResaltado();
    ultimaEtiqueta = "";
  }
}
```

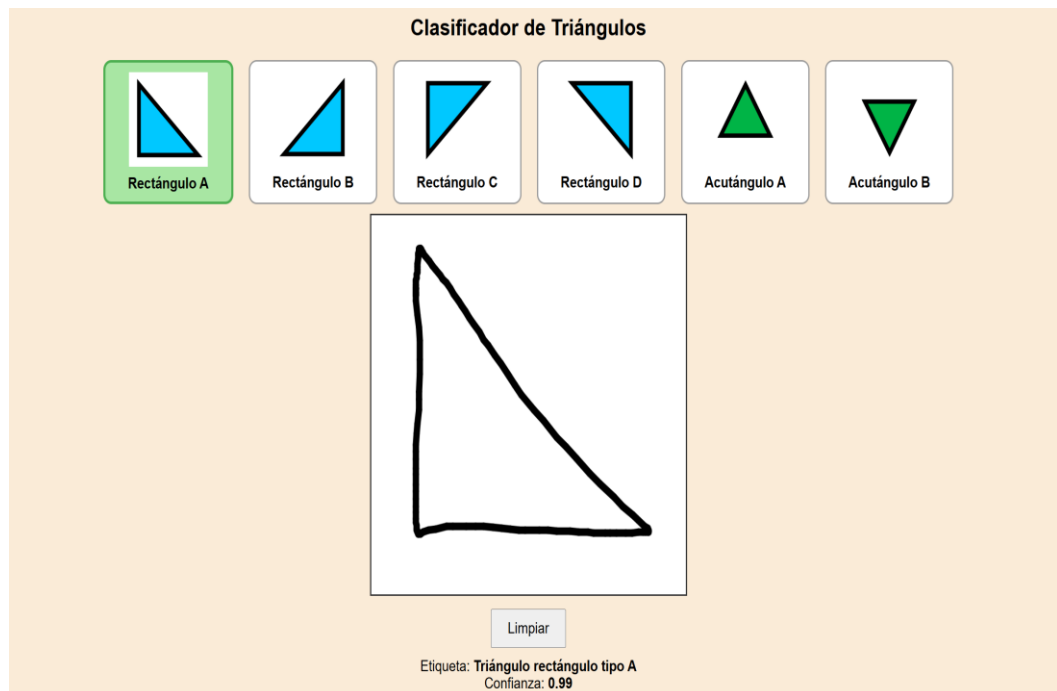
resaltarCasilla(etiqueta)


Aplico un estilo CSS para resaltar la casilla HTML correspondiente a la etiqueta del triángulo clasificado, permitiendo al usuario visualizar claramente la categoría detectada.

```
function resaltarCasilla(etiqueta) {
  quitarResaltado();
  const casilla = document.getElementById(etiqueta);
  if (casilla) {
    casilla.classList.add("activa");
  }
}
```

RESULTADOS





	[NOMBRE DEL TEMA DEL TRABAJO FINAL]	
	[APELLIDOS Y NOMBRES]	[ESCALA]

5. CONTROLAR

- **Verificar el cumplimiento de los procesos desarrollados en la propuesta de solución del caso práctico.**

EVIDENCIAS	CUMPLE	NO CUMPLE
• ¿Se identificó claramente la problemática del caso práctico?	<input type="checkbox"/>	<input type="checkbox"/>
• ¿Se desarrolló las condiciones de los requerimientos solicitados?	<input type="checkbox"/>	<input type="checkbox"/>
• ¿Se formularon respuestas claras y fundamentadas a todas las preguntas guía?	<input type="checkbox"/>	<input type="checkbox"/>
• ¿Se elaboró un cronograma claro de actividades a ejecutar?	<input type="checkbox"/>	<input type="checkbox"/>
• ¿Se identificaron y listaron los recursos (máquinas, equipos, herramientas, materiales) necesarios para ejecutar la propuesta?	<input type="checkbox"/>	<input type="checkbox"/>
• ¿Se ejecutó la propuesta de acuerdo con la planificación y cronograma establecidos?	<input type="checkbox"/>	<input type="checkbox"/>
• ¿Se describieron todas las operaciones y pasos seguidos para garantizar la correcta ejecución?	<input type="checkbox"/>	<input type="checkbox"/>
• ¿Se consideran las normativas técnicas, de seguridad y medio ambiente en la propuesta de solución?	<input type="checkbox"/>	<input type="checkbox"/>
• ¿La propuesta es pertinente con los requerimientos solicitados?	<input type="checkbox"/>	<input type="checkbox"/>
• ¿Se evaluó la viabilidad de la propuesta para un contexto real?	<input type="checkbox"/>	<input type="checkbox"/>

6. VALORAR

- Califica el impacto que representa la propuesta de solución ante la situación planteada en el caso práctico.

CRITERIO DE EVALUACIÓN	DESCRIPCIÓN DEL CRITERIO	PUNTUACIÓN MÁXIMA	PUNTAJE CALIFICADO POR EL ESTUDIANTE
Identificación del problema	Claridad en la identificación del problema planteado.	3	
Relevancia de la propuesta de solución	La propuesta responde adecuadamente al problema planteado y es relevante para el contexto del caso práctico.	8	
Viabilidad técnica	La solución es técnicamente factible, tomando en cuenta los recursos y conocimientos disponibles.	6	
Cumplimiento de Normas	La solución cumple con todas las normas técnicas de seguridad, higiene y medio ambiente.	3	
PUNTAJE TOTAL		20	

