



## Trabajo Práctico N°1

Materia: **IA4.4 Procesamiento De Imágenes**

Carrera: **Tecnicatura Universitaria en Inteligencia Artificial**

Primer integrante: **Gonzalo Uriel Calvo Rodriguez**

Segundo integrante: **Ignacio Eloy González**

Tercer Integrante: **Nicolás noir**

Fecha: 21/10/2024

**Universidad Nacional de Rosario**  
**Facultad de Ciencias Exactas, Ingeniería y Agrimensura**

# PROBLEMA 1:

## INTRODUCCIÓN

En este trabajo práctico, tiene como objetivo desarrollar una herramienta para la ecualización de una imagen para detectar caracteres ocultos

A través del procesamiento de imágenes se busca realizar una transformación local de ecualización del histograma

## ENUNCIADO

El programa debe procesar una imagen que consta de:

- La imagen es un cuadrado blanca con 5 cuadrados negros con diferentes valores dentro de estos cuadrados

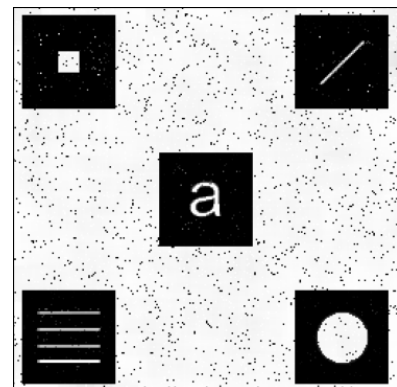
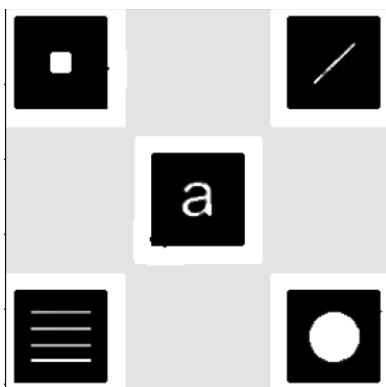
El objetivo es encontrar los detalles escondidos dentro de los cuadrados negros y hacerlos vizibles

## METODOLOGÍA RESOLUCIÓN

Procesamiento de imagen

En este proceso, realizamos una función que se le ingresa una imagen y el umbral que utilizaremos para procesarlo

1. Cargamos la imagen y luego realizamos un filtro de la mediana para sacar el ruido (salt and pepper), ya que después de ecualizar el histograma se observan puntos negros sobre toda la imagen.



```
imagen=cv2.imread('Imagen_con_detalles_escondidos.tif',cv2.IMREAD_GRAYSCALE)

imagen_estirada = cv2.medianBlur(imagen, 3)
```

2. Agregamos un borde para evitar posibles problemas con los índices fuera de rango y luego creamos una imagen de ceros.

```
imagen_estirada = cv2.copyMakeBorder(imagen_estirada,
tamano_ven//2,tamano_ven//2, tamano_ven//2,
tamano_ven//2,cv2.BORDER_REPLICATE)

imagen_equalizada = np.zeros_like(imagen)
```

3. Realizamos un bucle por el ancho y largo de la imagen, para aplicar la ecualización del histograma a lo largo y a lo ancho

```
for i in range(imagen.shape[0]):
    for j in range(imagen.shape[1]):

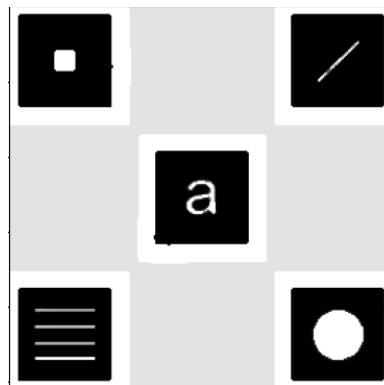
        local_window = imagen_estirada[i:i+tamano_ven, j:j+tamano_ven]

        local_equalized = cv2.equalizeHist(local_window)

        imagen_equalizada[i, j] = local_equalized[tamano_ven//2, tamano_ven//2]
```

## RESULTADO

1. **Imagen:** Una imagen con el histograma de la imagen ecualizado, mostrando el contenido de los recuadros, que previamente no se veían.



## PROBLEMA 2:

### INTRODUCCIÓN

En este trabajo práctico, tiene como objetivo desarrollar una herramienta automatizada para la corrección de exámenes en formato PNG.

A través del procesamiento de imágenes se busca extraer información importante de las imágenes y verificar el encabezado con los datos del alumno y evaluar las respuestas dadas.

### ENUNCIADO

El programa debe procesar un conjunto de imágenes donde cada imagen consta de:

- Un encabezado con información del alumno (nombre, fecha, clase).
- Un cuerpo de preguntas de opción múltiple con cuatro posibles respuestas marcadas manualmente por los estudiantes.

El objetivo es validar los datos en el encabezado y corregir las respuestas seleccionadas comparándolas con la clave correcta

### METODOLOGÍA RESOLUCIÓN

Procesamiento de imágenes

En este proceso utilizamos técnicas de de procesamiento de imágenes con la librería OpenCV

#### 1. Umbralización y binarización.

Aplicamos umbralización para convertir las imágenes de escala de grises en binarias, para facilitar la identificación de contornos y componentes conectados.

```
img = examen
img_bin_col = img<90
_, img_th1 = cv2.threshold(img,150,255,cv2.THRESH_BINARY)
```

En este caso utilizamos 2 tipos de umbralización y binarización, ya que al hora de detectar líneas horizontales y verticales obtuvimos resultados variados dependiendo el tipo de umbralización que utilizemos. En específico cuando buscamos líneas horizontales en la imagen, el cv2.threshold en ese umbral permite que se encuentren todas las líneas horizontales, evitando que obvie algunas ,y cuando buscamos líneas verticales la

umbralización por condición en ese umbral vimos que evitaba un problema de líneas de grosores de más de un pixel, como puede ser en los recuadros de la columna derecha.

Name: <u>JUAN PEREZ</u> Date: <u>11/07/24</u> Class: <u>1</u>	
1 The Earth's system that involves all our air is called the <u>C</u> A geosphere B hydrosphere C atmosphere D biosphere	6 The gaseous layers of the atmosphere are held to Earth's surface by <u>B</u> A their weight B gravity C the sun D none of the above

## 2. Detección de líneas de los recuadros.

Para la localización de los recuadros de preguntas y datos del alumno, se calculó la suma de valores en las columnas y filas de la imagen, encontrando los puntos de transición que corresponden a líneas de separación.

```
sumas_column = np.sum(img_bin_col, axis=0)
lineas_verticales = np.where(sumas_column >= (np.max(sumas_column)) * 0.8)[0]
sumas_fil = np.sum(img_th1, axis=1)
lineas_horiz = np.where(sumas_fil <= np.min(sumas_fil) * 2)[0][1:]
```

## 3. Detección del encabezado.

Definimos una función para encontrar el mayor componente horizontal de una imagen, que recorre todos los componentes conectados de la misma y encuentra el de mayor "ancho" y nos devuelve dicho valor y su centroide.

Luego utilizamos dicha función para detectar la coordenada central del encabezado.

```
encabezado = img_th1[0:lineas_horiz[0]-3, 0:lineas_verticales[-1]]
_, cord_central_guiones_encabezado = encontrar_mayor_componente_horizontal(encabezado)
```

#### 4. Detección de renglones del encabezado.

Para verificar la cantidad de caracteres en el nombre, fecha y clase del encabezado, se utilizó la técnica de componentes conectados, identificando las regiones de píxeles conectados que representaban el texto.

```
cant_caracteres_nombre , _ = cv2.connectedComponents(nombre_examen_alt,
connectivity=8)
cant_caracteres_fecha, _ = cv2.connectedComponents(fecha_examen_alt,
connectivity=8)
cant_caracteres_clase, _ = cv2.connectedComponents(clase_examen_alt,
connectivity=8)
```

#### 5. Detección de los renglones de respuesta.

Recorremos todos los recortes de los recuadros de la respuesta y utilizamos la función encontrar\_mayor\_componente\_horizontal para obtener las coordenadas de su centroide y el valor del ancho. Luego recortamos la imagen mediante el uso de estas coordenadas y un valor definido por nosotros de 14 píxeles de x. Por último invertimos la imagen binaria del recorte de la respuesta

```
for respuesta in range(len(recuadros_lista)):
    ancho_maximo, cord_central =
    encontrar_mayor_componente_horizontal(recuadros_lista[respuesta])
    cord_linea =
    recuadros_lista[respuesta][int(cord_central[1])-14:int(cord_central[1]),
    int(cord_central[0])-int(ancho_maximo//2):int(cord_central[0])+int(ancho_maxim
o//2)]

    cord_linea_alt = cv2.bitwise_not(cord_linea)
```

#### 6. Detección de letras en la respuesta.

Primero encontramos la cantidad de componentes conectados en la respuesta para poder analizar la cantidad de letras en la respuesta. Luego encontramos los contornos de los recortes del renglón de la respuesta para poder analizar qué letra es la que fue respondida.

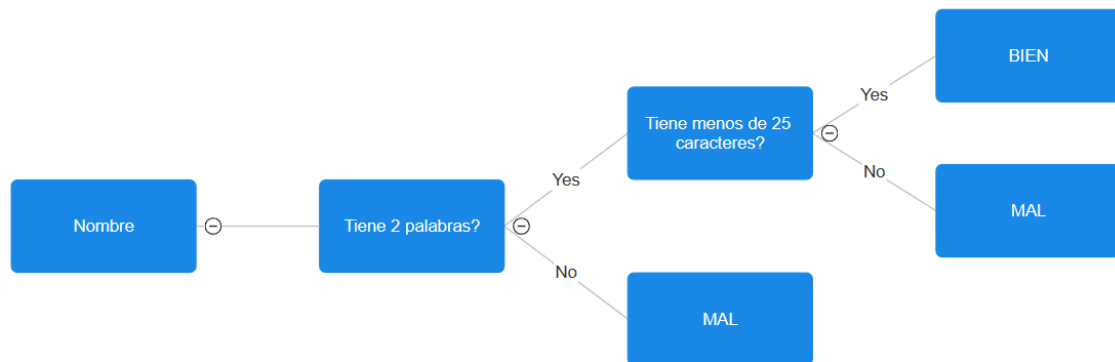
```
num_labels, _, _, _ = cv2.connectedComponentsWithStats(cord_linea_alt,
connectivity=8)

contornos, _ = cv2.findContours(cord_linea_alt,cv2.RETR_LIST
,cv2.CHAIN_APPROX_NONE)
```

## 7. Validación de los datos de encabezado.

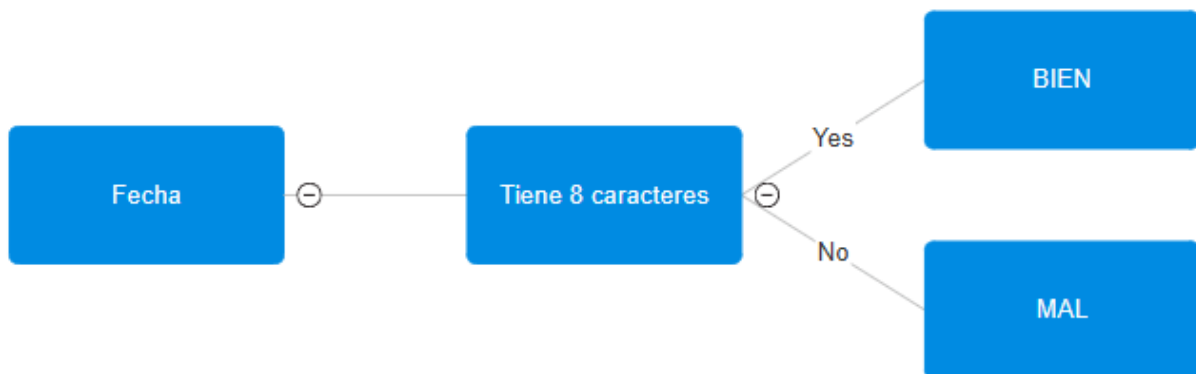
### a. Nombre:

Para validar el nombre seguiremos el siguiente árbol de decisiones:



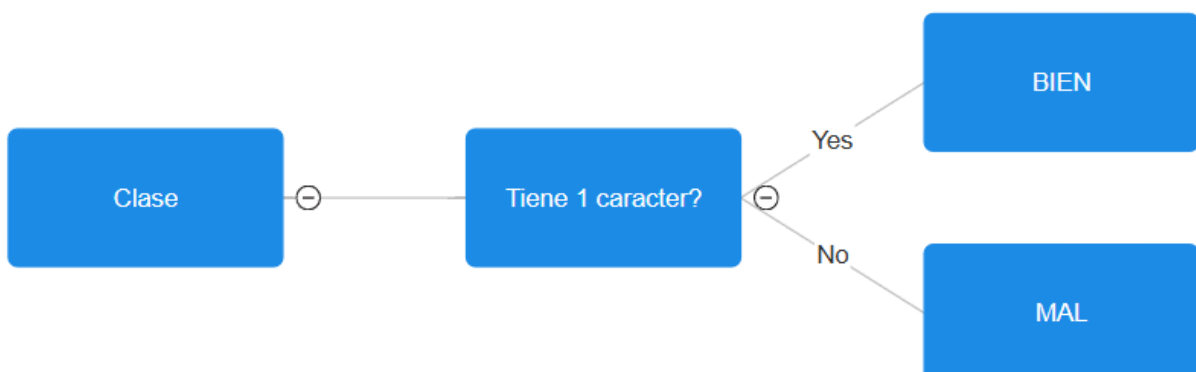
### b. Fecha:

Para validar la fecha seguiremos el siguiente árbol de decisiones:



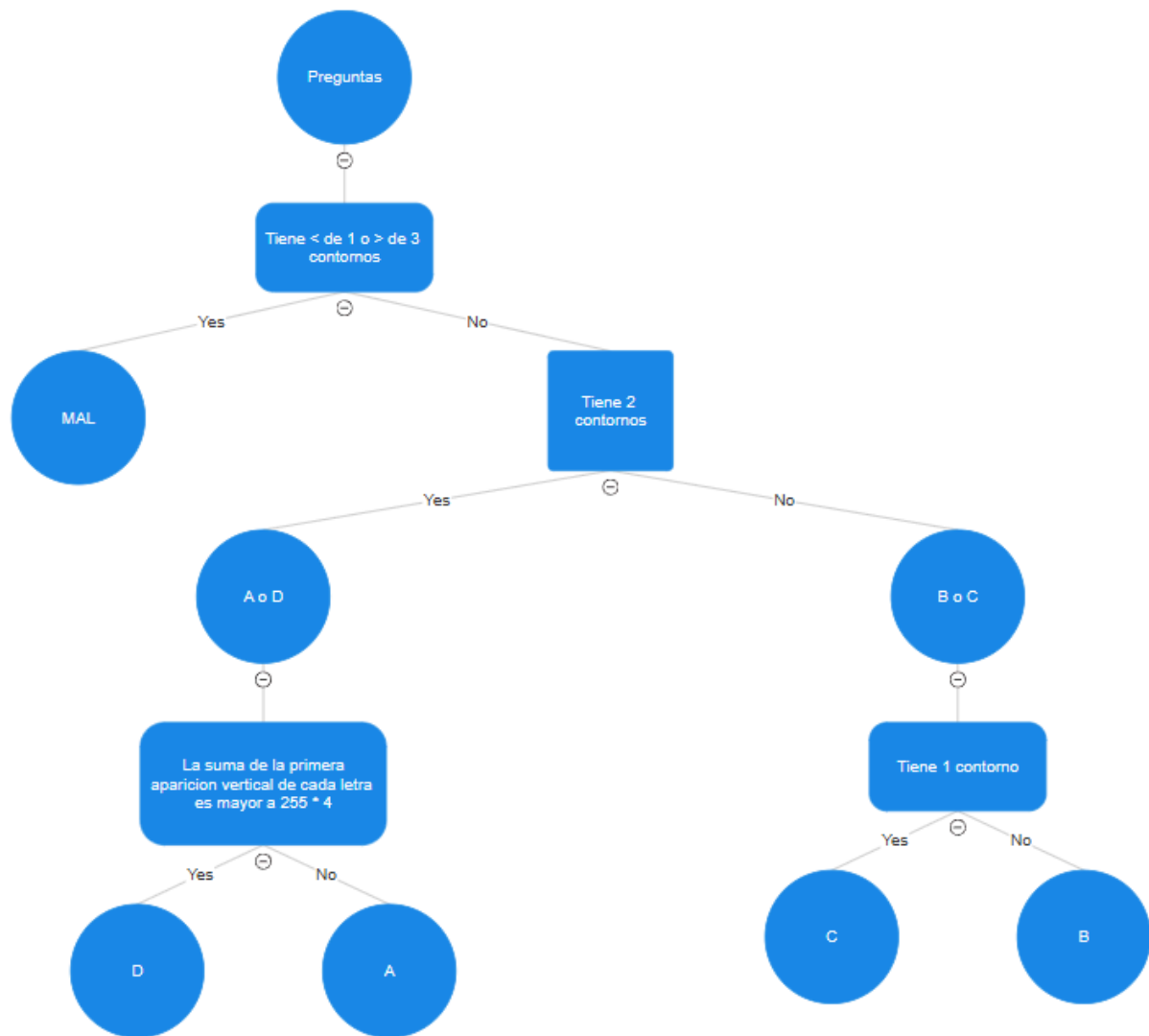
### c. Clase:

Para validar la clase seguiremos el siguiente árbol de decisiones:



## 8. Corrección de las respuestas del examen.

Para corregir las preguntas seguiremos el siguiente árbol de decisiones:

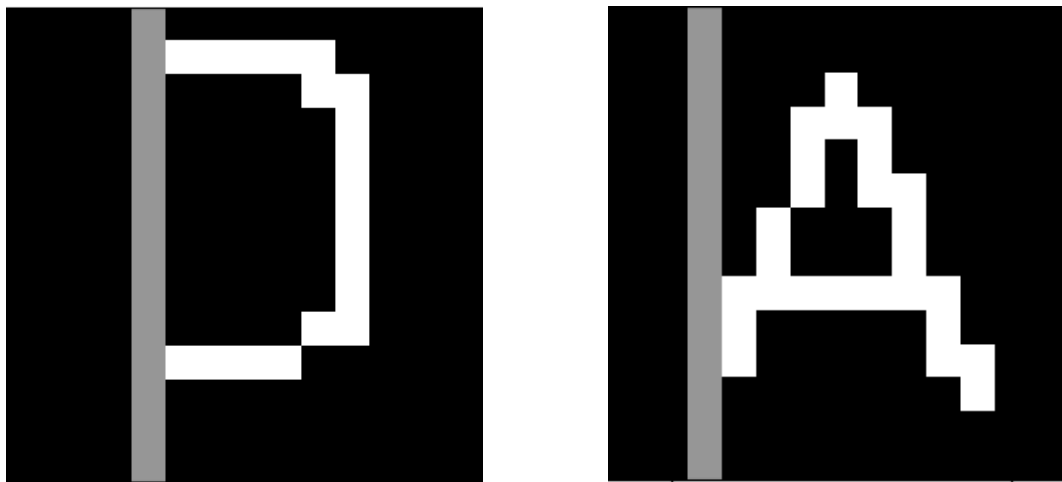




### 9. Análisis de las letras para el reconocimiento de las mismas.

Una vez seleccionadas las condiciones para la validación y corrección, necesitamos identificar las letras que fueron respuestas para la corrección de las preguntas. Para eso utilizaremos los contornos ya encontrados en el punto 6 de la detección de las letras, esto nos permite diferenciar entre B, C y A o D, es decir, A y D tienen la misma cantidad de contornos, un contorno padre y un hijo, y por otro lado B y C tienen 1 y 3 contornos respectivamente, lo cual los hace diferenciables.

Para poder diferenciar A y D utilizamos la imagen donde la letra tiene valores 255 y el fondo 0, sumamos las columnas del recorte y en la primera coordenada donde se encuentre un valor diferente a 0 observamos la suma de valores 255, siendo que la letra D tiene la columna de la altura de la letra completa de valores 255, entonces su suma supera el umbral de  $255 \times 5$  (5 siendo los píxeles), ya que tiene 7 u 8 píxeles de alto.



### 10. Creación de la imagen de salida con el informe de aprobados

Para crear esta imagen lo primero que tenemos que deducir es el ancho y alto de la misma, para esto observamos cuáles eran los anchos de los recortes de los nombres y si eran todos iguales, lo cual era verdadero, entonces utilizamos el ancho del renglón del nombre y el alto del renglón del nombre multiplicado por la cantidad de imágenes de exámenes. Además a este ancho de la imagen le agregamos 30 píxeles los cuales van a ser el espacio para mostrar su condición en el examen. Creamos 2 imágenes, una con 3 canales para poder pintar la condición y una con 1 canal para poder encontrar el centroide del recuadro de la condición.

```
alto_img = nombre_examen.shape[0]*(len(imagenes))
ancho_img = nombre_examen.shape[1]
imagen_respuestas_alumnos = np.ones((alto_img, ancho_img+30, 3), np.uint8)*255
imagen_respuestas_alumnos_un_canal =
np.zeros((alto_img, ancho_img+30), np.uint8)
```

Para informar la condición de cada alumno, pensamos en hacer círculos rojos para desaprobados y círculos verdes para aprobados. Para hacer estos círculos en la imagen es necesario encontrar el centro del espacio donde va a ser ubicado el círculo, es decir, el centro de los 30 píxeles que agregamos al ancho.

Este centro lo vamos a encontrar recortando el recuadro donde va a ir este círculo y usando `connectedComponentsWithStats()`, ya que nos dará el centroide del fondo, por ser el único componente que tiene el recorte. También pensamos en que la imagen debería tener un formato de planilla, entonces por cada nombre de alumno hicimos la línea horizontal superior que diferencie los mismos.

```
indx_img = 0
for _, examen_evaluado_value in examenes.items():
    imagen_respuestas_alumnos[examen_evaluado_value[2].shape[0]*
    indx_img:examen_evaluado_value[2].shape[0]*(indx_img+1),0:
    examen_evaluado_value[2].shape[1]] =
    cv2.cvtColor(examen_evaluado_value[2],cv2.COLOR_GRAY2RGB)

    cv2.line(imagen_respuestas_alumnos,(0,examen_evaluado_value[2].shape[0]*
    indx_img),(imagen_respuestas_alumnos.shape[1],
    examen_evaluado_value[2].shape[0]*(indx_img)),(0,0,0),1,cv2.LINE_4)

    recuadro_condicion =
    imagen_respuestas_alumnos_un_canal[examen_evaluado_value[2].shape[0]*indx_img:
    examen_evaluado_value[2].shape[0]*(indx_img+1),ancho_img:
    imagen_respuestas_alumnos.shape[1]]
```

Una vez que ya tenemos cada recorte buscamos su centroide para usarlo de coordenada central de nuestro círculo.

Una vez obtenido el centroide procedemos a verificar la condición del examen y dibujar un círculo verde o rojo de un radio de 7 píxeles.

Desglosando la línea donde dibujamos dicho círculo, `cv2.circle` recibe de parámetros: La imagen donde se va a dibujar dicho círculo, un punto como tuplas con coordenadas inicio/fin, el radio del círculo, el color del círculo y si va a estar lleno o vacío dicho círculo.

el `(centroide_condicin[0][0])+ancho_img` es la coordenada en y del centroide, como nosotros calculamos el centroide de manera al querer dibujarlo en la imagen completa hacemos un corrimiento en y sumándole los píxeles que le faltan.

`centroide_condicin[0][1]` es la coordenada en x del centroide, a la cual le sumamos `examen_evaluado_value[2].shape[0]` que es el tamaño en x de cada nombre multiplicado por `indx_img` que es un valor auto-incremental con la cantidad de nombres.

```

_, _, _, centroide_condicion =
cv2.connectedComponentsWithStats(recuadro_condicion)

if examen_evaluado_value[1]['Condicion'] == 'A':
cv2.circle(imagen_respuestas_alumnos, (int(centroide_condicion[0][0])+ancho_img
,int(centroide_condicion[0][1]+examen_evaluado_value[2].shape[0]*indx_img)+1),
7, (0,255,0), -1)
else:

cv2.circle(imagen_respuestas_alumnos, (int(centroide_condicion[0][0])+ancho_img
,int(centroide_condicion[0][1]+examen_evaluado_value[2].shape[0]*indx_img)+1),
7, (255,0,0), -1)

indx_img += 1

```

Finalmente hicimos la última línea, que es la vertical, de la planilla para diferenciar nombres y condiciones. Esto gracias a que el ancho que tenemos guardado es del recorte del nombre y no del ancho total, lo que facilita marcar esta línea.

```

cv2.line(imagen_respuestas_alumnos, (ancho_img,0),
(ancho_img,imagen_respuestas_alumnos.shape[0]), (0,0,0), 1, cv2.LINE_4)

```

## RESULTADO

El sistema procesa y devuelve la corrección de los exámenes en un formato de diccionario. Para cada examen, se obtienen los siguientes resultados:

1. **Validación del Encabezado:** Los datos del alumno se etiquetan como "OK" o "MAL", dependiendo de si cumplen con las reglas establecidas.

```






Name: MAL
Date: OK
Class: OK

```

2. **Corrección de Preguntas:** Se corrigen las respuestas de las preguntas de opción múltiple, devolviendo "OK" si la respuesta es correcta, o "MAL" en caso contrario. El número del examen coincide con la ubicación de los nombres en la imagen de salida.

```
Correcciones del examen 2
Pregunta 1: MAL
Pregunta 2: MAL
Pregunta 3: MAL
Pregunta 4: OK
Pregunta 5: MAL
Pregunta 6: OK
Pregunta 7: OK
Pregunta 8: MAL
Pregunta 9: MAL
Pregunta 10: OK
```

3. **Imagen:** Una imagen generada con el “crop” de los nombres de cada uno de los exámenes y una diferenciación entre aprobado y desaprobado.

ESTEBANALVAREZ	
MARIA	
MARIA LOPEZ	
LUCAS FERNANDEZ	
JUAN PEREZ	

# CONCLUSIÓN

Para dar por terminado este trabajo práctico nos gustaría abordar las dificultades con algunos puntos específicos de las consignas, como la diferenciación entre letras y el manejo adecuado de las coordenadas. Estos aspectos nos llevaron a realizar varias modificaciones en el código debido a errores encontrados durante el desarrollo.

Pese a estos inconvenientes, los resultados finales son gratificantes, ya que logramos superarlos y cumplir con la consigna de manera satisfactoria. A lo largo del proceso, que incluyó largos periodos de prueba y error, pudimos aprovechar la experiencia para generar un gran aprendizaje sobre las primeras unidades de Procesamiento de Imágenes.

Por otro lado, notamos una marcada diferencia en la dificultad entre los problemas planteados. El primer problema, relacionado con la ecualización del histograma, no era muy complicado, pero presentaba desafíos como el ruido tipo "pepper" en la imagen y la comprensión del tamaño de la ventana de ecualización. En contraste, el segundo problema fue más complejo, ya que abordaba múltiples puntos de las unidades estudiadas. Cada paso requería decisiones importantes, como al umbralizar la imagen para detectar líneas verticales, lo cual no siempre funcionaba bien para las líneas horizontales, generando complicaciones al buscar las coordenadas o separar elementos.

Asimismo, otro desafío fue detectar renglones dentro de un recuadro con palabras, lo que no se resolvía con los métodos anteriores, obligándonos a usar componentes conectados y comparar anchos. El problema final, y el más problemático, fue diferenciar letras. Aunque detectar la cantidad de letras resultó sencillo con los componentes conectados, diferenciarlas entre sí fue complicado, ya que algunas compartían contornos similares, lo que agregó más decisiones al proceso. Optamos por una solución poco robusta, comparando la suma de la primera columna de las letras A y D.

Finalmente, generar la imagen con los recortes de los nombres y condiciones fue menos complejo, pero presentó su propio conjunto de desafíos, ya que no involucra técnicas o métodos previamente utilizados.