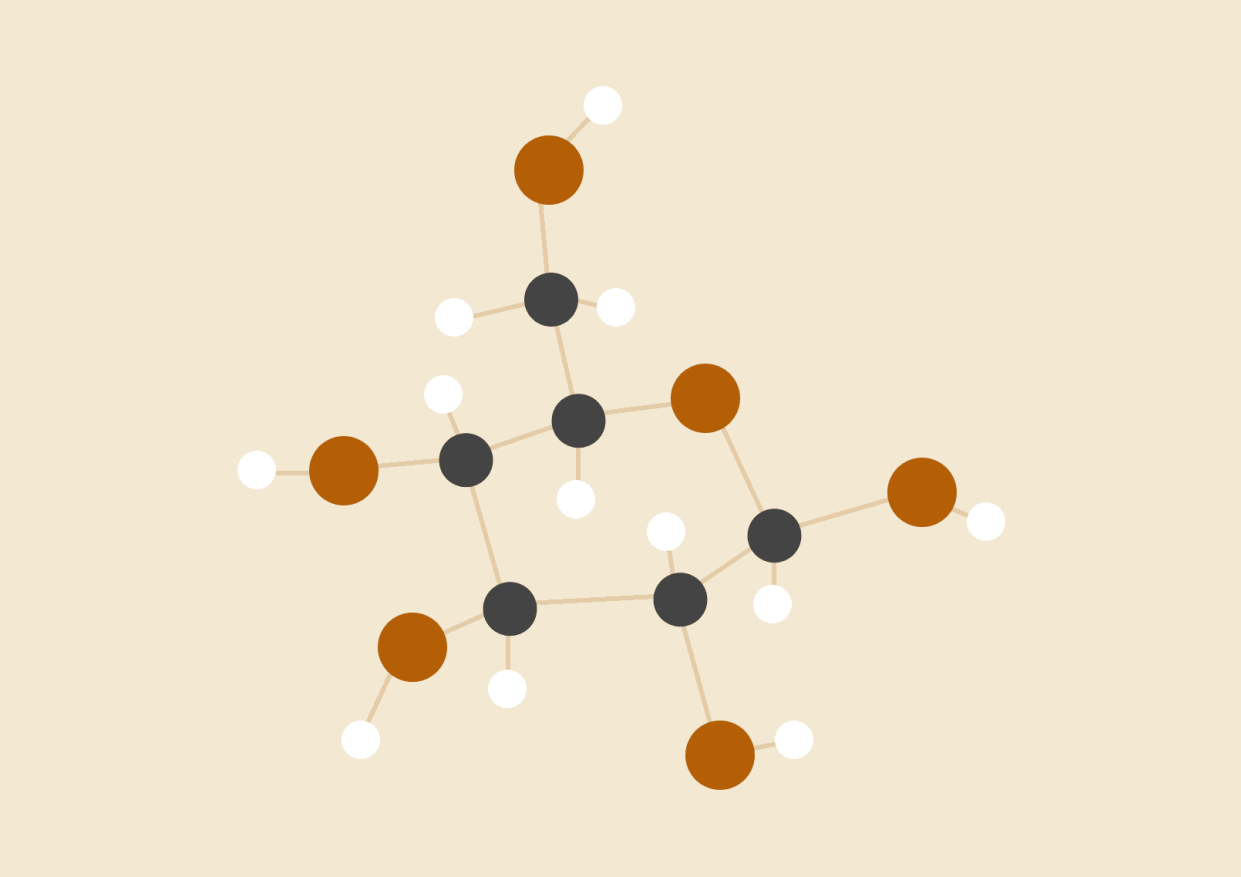
TRABAJO PRÁCTICO INTEGRADOR

IA4.4 Procesamiento de Imágenes I

Tecnicatura Universitaria en Inteligencia Artificial



Brizuela Cipolletti Sofía

Fontana Gustavo Julián

01/10/2023
Universidad Nacional de Rosario
Facultad de Ciencias Exactas, Ingeniería y Agrimensura

INTRODUCCIÓN

En primer lugar se solicita la implementación de una función que permita realizar una ecualización local del histograma para procesar una imagen, mejorando su contraste y haciendo posible el resaltado de detalles que posean intensidades similares al fondo de la imagen.

Para la ecualización se utiliza una venta de tamaño constante que se desplaza sobre la imagen realizando la transformación localmente.

Por último se pretende validar una serie de formularios en formato de imagen. Para realizar esto es necesario la detección de las celdas en donde se guarda la información relevante, para luego detectar los posibles caracteres y palabra en cada uno de los campo que cumplan con las siguientes condiciones:

1. Nombre y apellido: Debe contener al menos 2 palabras y no más de 25 caracteres en total.

2. Edad: Debe contener 2 o 3 caracteres.

3. Mail: Debe contener 1 palabra y no más de 25 caracteres.

4. Legajo: 8 caracteres formando 1 sola palabra.

5. Preguntas: se debe marcar con 1 caracter una de las dos celdas SI y NO. No pueden estar ambas vacías ni ambas completas.

6. Comentarios: No debe contener más de 25 caracteres.

PROCEDIMIENTO

- **Ejercicio 1:**

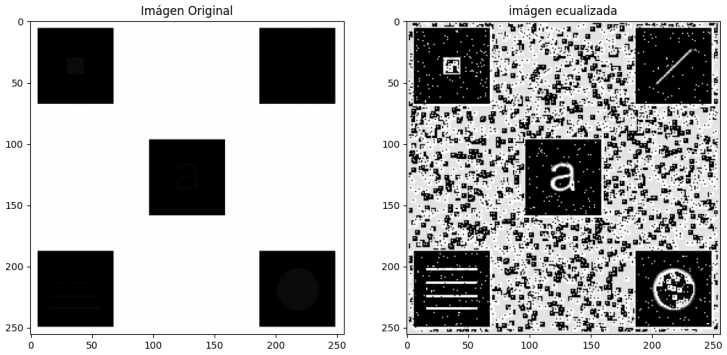
La función ‘ecualizacion_local_histograma’ recibe como parámetros a la imagen a procesar y el tamaño de la ventana. Se obtiene el tamaño de la imagen (ancho y alto), la mitad del tamaño de la ventana y se crea una copia de la imagen original para retornarla luego de aplicar la transformación.

Para desplazar la ventana sobre la imagen e ir realizando la ecualización, se itera sobre sobre los ejes x e y en un rango elegido apropiadamente para cada eje, con el fin de evitar que la ventana se desplace fuera de las fronteras de la imagen.

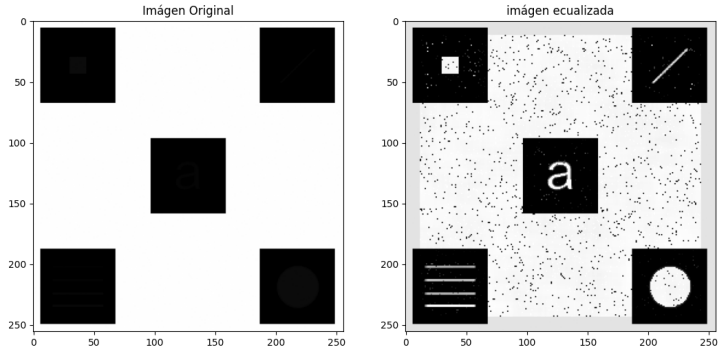
En cada iteración, se obtienen las coordenadas de la ventana, se las almacena en una variable a la cual luego se le calcula su histograma,se le realiza la ecualización, y por ultimo, en la copia de la imagen original se remplaza en la zona de las coordenadas actuales con la transformación realizada.

Una vez que la ventana se desplazó por todo el rango de la imagen realizando una transformación local se retorna la copia de la imagen ya procesada.

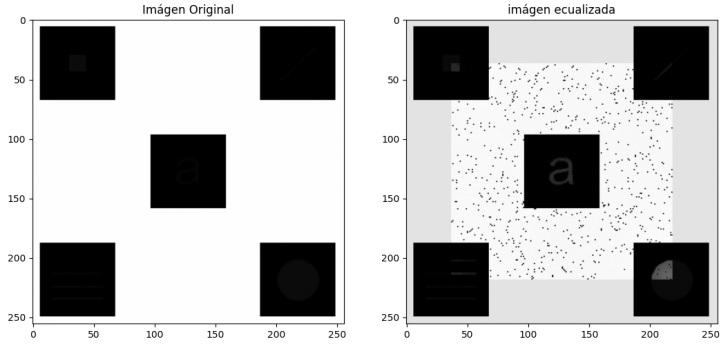
SALIDAS EJ 1:



Ecualización con una ventana de tamaño 5x5



Ecualización con una ventana de tamaño 25x25



Ecualización con una ventana de tamaño 75x75

Se puede notar que en tamaños mas chicos de la ventana se detecta mucha variación local en la intensidad de los grises debido al ruido, en cambio cuando el tamaño aumenta se reduce el ruido y empieza a disminuir la capacidad de detectar detalles.

En un punto medio entre ambos se mejora el contraste de la imagen original y se puede apreciar los detalles sin la interferencia de tanto ruido.

- **Ejercicio 2:**

Para la validación de los formulario se utilizan cuatro funciones para ir procesando las imágenes y obteniendo lo necesario en cada paso.

La primer función usada ‘extraer_campos’ permite extraer las regiones de interés realizando una binarización de la imagen con el fin de mejorar la detección de bordes. Con un umbral adecuado se logra tener las lineas delimitadoras con el grosor de un pixel. Ademas, se suman los valores de los pixeles en cada columna y en cada fila, y se establece un umbral para las filas y columnas con el que se logra detectarlas.

Las coordenadas de filas y columnas son guardadas cada una en una variable. A continuación, se crea un diccionario para guardar los indices de cada campo relevante del formulario y las coordenadas de cada uno de los renglones, utilizando para ello las coordenadas de las filas almacenadas previamente. (las coordenadas de cada renglón están dadas por la coordenada de una fila y su inmediata siguiente)

La función retorna un diccionario con los campos de interés del formulario y sus coordenadas.

La segunda función ‘contar_caracteres_palabras’ cuenta los caracteres y el total de palabras en la región de interés. Para eso recibe como argumento un array (stats, que resulta de detectar componentes conectadas) con coordenadas y un rango inicial y final sobre el cual contar caracteres y palabras.

El array que ingresa a la función es ordenado en base a su primer indice, que corresponde a la coordenada x inicial del bounding box; se eliminan los dos primeros valores del array que resultan de la detección de los restos de los bordes al momento de separar los renglones.

Se inicializan variables para almacenar el total de caracteres, palabras y una que va a almacenar las coordenadas de las x finales de cada bounding box siempre que se cumpla con la distancia mínima establecida para diferenciar un espacio entre caracteres y un espacio entre palabras.

Luego se inician las condiciones en los rangos tomados como argumento, y si se cumplen para cada caso particular se incrementan las variables inicializadas según corresponda.

La función retorna el total de caracteres y palabras en cada región de interés.

La tercer función ‘validar_si_no’ calcula la cantidad de caracteres en un renglón con dos columnas. Recibe como argumento una imagen y una lista con tuplas de coordenadas por cada columna. Se obtienen las componentes conectadas sobre la región de interés en la imagen, calculada con las coordenadas para cada una de las columnas.

Además se hace un slicing sobre stats para quitar los valores de los bounding box correspondientes a los restos de los bordes de la región de interés en caso de que los hubiera.

Se hace una instancia de la cantidad de caracteres llamando a la función ‘contar_caracteres_palabras’ para cada una de las columnas de la región de interés, y luego se los retorna.

La cuarta función ‘validar_formulario’ recibe una imagen como argumento y verifica en cada uno de los campos de la imagen (formulario) si cumplen con las condiciones impuestas.

En primer lugar se almacena el resultado de llamar a la función ‘extraer_campos’ sobre el cual luego se va a iterar (campos y coordenadas). Además, se crea un diccionario con los campos del formulario como claves y sus valores correspondientes vacíos.

Al iniciar el ciclo se binariza la imagen con un umbral adecuado para detectar correctamente el contenido. Luego, se van a iniciar los condicionales sobre los que van a entrar cada uno de los campos de la imagen (formulario) . Para los campos en que el contenido se encuentra en una sola columna se van a detectar las componentes conectadas en la región de interés de la imagen sobre las coordenadas actuales de la iteración. Se van a obtener la cantidad de caracteres y palabras invocando a la función ‘contar_caracteres_palabras’, y se va a modificar el valor del campo actual del diccionario en base al resultado de la validación.

En el caso de los campo con dos columnas (Preguntas 1, 2 y 3. SI/NO) la validación se va a realizar invocando a la función ‘validar_si_no’ que va a retornar la cantidad de caracteres en cada columna. Con estos resultados, se verifican las condiciones impuestas y se modifica el diccionario acorde al resultado.

Por último se retorna la validación del formulario, con un ‘OK’ si el contenido del campo es correcto y un ‘MAL’ en caso contrario.

SALIDAS EJ 2:

```
>>> validar_formulario(f0)
{'Nombre y Apellido': 'MAL',
'Edad': 'MAL',
'Mail': 'MAL',
'Legajo': 'MAL',
'Pregunta 1': 'MAL',
'Pregunta 2': 'MAL',
'Pregunta 3': 'MAL',
'Comentarios': 'MAL'}
```

```
>>> validar_formulario(f1)
{'Nombre y Apellido': 'OK',
'Edad': 'OK',
'Mail': 'OK',
'Legajo': 'OK',
'Pregunta 1': 'OK',
'Pregunta 2': 'OK',
'Pregunta 3': 'OK',
'Comentarios': 'OK'}
```

```
>>> validar_formulario(f2)
{'Nombre y Apellido': 'MAL',
'Edad': 'MAL',
'Mail': 'MAL',
'Legajo': 'MAL',
'Pregunta 1': 'MAL',
'Pregunta 2': 'MAL',
'Pregunta 3': 'MAL',
'Comentarios': 'MAL'}
```

```
>>> validar_formulario(f3)
{'Nombre y Apellido': 'OK',
'Edad': 'OK',
'Mail': 'OK',
'Legajo': 'OK',
'Pregunta 1': 'OK',
'Pregunta 2': 'OK',
'Pregunta 3': 'OK',
'Comentarios': 'OK'}
```

```
>>> validar_formulario(f4)
{'Nombre y Apellido': 'MAL',
'Edad': 'MAL',
'Mail': 'MAL',
'Legajo': 'MAL',
'Pregunta 1': 'OK',
'Pregunta 2': 'MAL',
'Pregunta 3': 'MAL',
'Comentarios': 'MAL'}
```

```
>>> validar_formulario(f5)
{'Nombre y Apellido': 'OK',
'Edad': 'OK',
'Mail': 'OK',
'Legajo': 'OK',
'Pregunta 1': 'MAL',
'Pregunta 2': 'MAL',
'Pregunta 3': 'MAL',
'Comentarios': 'OK'}
```

APÉNDICE A

#EJERCICIO 1:

```
#Función de ecualización local del histograma
def ecualizacion_local_histo(imagen, tamaño_ventana):
    altura, ancho = imagen.shape
    imagen_de_salida = np.copy(imagen) #Imágen de salida (ecualizada)
    mitad_ventana = tamaño_ventana[0] // 2

    for y in range(mitad_ventana, altura - mitad_ventana):
        for x in range(mitad_ventana, ancho - mitad_ventana):
            window = imagen[y - mitad_ventana:y + mitad_ventana, x - mitad_ventana:x + mitad_ventana]
            histograma = cv2.calcHist([window], [0], None, [256], [0, 256])
            histograma_eq = cv2.equalizeHist(window)
            imagen_de_salida[y, x] = histograma_eq[mitad_ventana, mitad_ventana]
    return imagen_de_salida
```

#EJERCICIO 2

```
def extraer_campos(imagen):
    """Permite separar campos en una imagen de interes y guardar sus coordenadas
    en un dicionario."""
    umbral = 100
    imagen_binaria = np.where(imagen >= umbral, 1, 0)

    # Suma valores de píxeles a lo largo de las columnas y filas
```

```
suma_columnas = np.sum(imagen_binaria, axis=0)
suma_filas = np.sum(imagen_binaria, axis=1)

# Define umbrales para detectar líneas en columnas y filas
umbral_columnas = 0.7 * np.max(suma_columnas)
umbral_filas = 0.5 * np.max(suma_filas)

# Detectar líneas en columnas y filas
lineas_columnas = np.where(suma_columnas < umbral_columnas)[0]
lineas_filas = np.where(suma_filas < umbral_filas)[0]

#Extraigo los renglones
renglon = {}
for i in range(len(lineas_filas)-1):
    if i >= 1 and i <= 4:
        renglon[i] = (lineas_filas[i], lineas_filas[i+1], lineas_columnas[1], lineas_columnas[3])
    elif i >= 6 and i <= 8:
        renglon[i] = [(lineas_filas[i], lineas_filas[i+1], lineas_columnas[1], lineas_columnas[2]),
                (lineas_filas[i], lineas_filas[i+1], lineas_columnas[2], lineas_columnas[3])]
    elif i == 9:
        renglon[i] = (lineas_filas[i], lineas_filas[i+1], lineas_columnas[1], lineas_columnas[3])
renglon = {
    0: renglon[1], #NOMBRE Y APELLIDO
    1: renglon[2], #EDAD
    2: renglon[3], #MAIL
```

```
        3: renglon[4], #LEGAJO
        4: renglon[6], #PREGUNTA 1
        5: renglon[7], #PREGUNTA 2
        6: renglon[8], #PREGUNTA 3
        7: renglon[9]  #COMENTARIOS
    }
    return renglon
#-------------------------------------------------------------------------------------------------------------------#
def contar_caracteres_palabras(array, rango_inicial, rango_final):
    """Permite  calcular  la cantidad de caracteres y palabras de una imagen.
    Recibe como argumento un array con coordenadas de los boundign box de los
    caracteres y rango incicial y final sobre el cual establecer la condición
    de detección de caracteres y palabras."""
    #Ordeno el array en base a la primer coordenada (x inicial)
    array = array[array[:,0].argsort()]
    #Inicializo variables para guardar el total de caracteres y palabras
    x_finales = []
    caracteres = 0
    palabras = 0
    posicion_x_final = -1
    #Itero y actualizo las variables inicializadas si es condición
    if len(array) > rango_inicial and len(array) <= rango_final:
        palabras += 1
        for stat in array:
            x_finales.append(stat[0]+stat[2])
```

```
            caracteres += 1
        x_finales = x_finales[:-1]
        for st in array[1:]:
            posicion_x_final += 1
            if abs(st[0] - x_finales[posicion_x_final] ) > 7:
                palabras += 1
                #caracteres += 1 #Por si es necesario contar espacios
    return (caracteres,palabras)


def validar_si_no(img,coordenadas):
    """Permite validar campos de opcíon doble retornado cantidad de carecteres para cada opción.
    Recibe como argumento una imagen binaria y una lista de dos tuplas con coordenadas."""
    y1,y2,x1,x2 = coordenadas[0]
    num_labels_0, labels_0, stats_0, centroids_0 = cv2.connectedComponentsWithStats(img[y1:y2,x1:x2],
8, cv2.CV_32S)
    stats_0 = stats_0[2:]
    cant_caracteres_0, cant_palabras_0 = contar_caracteres_palabras(stats_0,0,1)
    y_1,y_2,x_1,x_2 = coordenadas[1]
    num_labels_1, labels_1, stats_1, centroids_1 =
cv2.connectedComponentsWithStats(img[y_1:y_2,x_1:x_2], 8, cv2.CV_32S)
    stats_1 = stats_1[2:]
    cant_caracteres_1, cant_palabras_1 = contar_caracteres_palabras(stats_1,0,1)
    return (cant_caracteres_0,cant_caracteres_1)
#-------------------------------------------------------------------------------------------------------------------------#
```

```
def validar_formulario(img_formulario):
    """Permite validar un formulario en base a las condiciones solicitadas. Retorna un
    diciconario con el nombre de cada campo y el valor 'OK' o 'MAL' según corresponda.
    Recibe como argumento la imagen del formulario a procesar."""
    #Se crea una instancia de un formulario llamando a la función 'extraer_campos'
    diccionario_formulario = extraer_campos(img_formulario)
    formulario = {'Nombre y Apellido':None,
                  'Edad':None,
                  'Mail':None,
                  'Legajo':None,
                  'Pregunta 1':None,
                  'Pregunta 2':None,
                  'Pregunta 3':None,
                  'Comentarios':None}
    #Se itera sobre sobra las claves y valores del diccionario
    for indice_campo, coordenada in diccionario_formulario.items():
        #Se convierte la imagen a binaria
        imagen = np.where(img_formulario >= 128, 0, 1); imagen = cv2.convertScaleAbs(imagen)
        if indice_campo == 0: #NOMBRE Y APELLIDO
            y1,y2,x1,x2 = coordenada
            num_labels, labels, stats, centroids = cv2.connectedComponentsWithStats(imagen[y1:y2,x1:x2],
8, cv2.CV_32S)
            stats = stats[2:]
            cant_caracteres_0, cant_palabras_0 = contar_caracteres_palabras(stats,0,25)
            formulario['Nombre y Apellido'] = 'OK' if cant_caracteres_0 <= 25 and cant_palabras_0 >= 2 else
```

```
'MAL'

        elif indice_campo == 1: #EDAD

            y1,y2,x1,x2 = coordenada

            num_labels, labels, stats, centroids = cv2.connectedComponentsWithStats(imagen[y1:y2,x1:x2], 8, cv2.CV_32S)

            stats = stats[2:]

            cant_caracteres_1, cant_palabras_1 = contar_caracteres_palabras(stats,0,3) #cant_palabras <= 2 si se considera válido al espacio

            formulario['Edad'] = 'OK' if cant_caracteres_1 > 0 and cant_caracteres_1 < 4 and cant_palabras_1 <= 1 else 'MAL'

        elif indice_campo == 2: #MAIL

            y1,y2,x1,x2 = coordenada

            num_labels, labels, stats, centroids = cv2.connectedComponentsWithStats(imagen[y1:y2,x1:x2], 8, cv2.CV_32S)

            stats = stats[2:]

            cant_caracteres_2, cant_palabras_2 = contar_caracteres_palabras(stats,0,25)

            formulario['Mail'] = 'OK' if cant_caracteres_2 > 0 and cant_caracteres_2 <= 25 and cant_palabras_2 == 1 else 'MAL'

        elif indice_campo == 3: #LEGAJO

            y1,y2,x1,x2 = coordenada

            num_labels, labels, stats, centroids = cv2.connectedComponentsWithStats(imagen[y1:y2,x1:x2], 8, cv2.CV_32S)

            stats = stats[2:]

            cant_caracteres_3, cant_palabras_3 = contar_caracteres_palabras(stats,0,8)

            formulario['Legajo'] = 'OK' if cant_caracteres_3 == 8 and cant_palabras_3 == 1 else 'MAL'

        elif indice_campo == 4: #PREGUNTA1

            si, no = validar_si_no(imagen,coordenada)
```

```
            formulario['Pregunta 1'] = 'OK' if (si == 1) ^ (no == 1) else 'MAL'
        elif indice_campo == 5: #PREGUNTA2
            si, no = validar_si_no(imagen,coordenada)
            formulario['Pregunta 2'] = 'OK' if (si == 1) ^ (no == 1) else 'MAL'
        elif indice_campo == 6: #PREGUNTA3
            si, no = validar_si_no(imagen,coordenada)
            formulario['Pregunta 3'] = 'OK' if (si == 1) ^ (no == 1) else 'MAL'
        elif indice_campo == 7: #COMENTARIOS
            y1,y2,x1,x2 = coordenada
            num_labels, labels, stats, centroids = cv2.connectedComponentsWithStats(imagen[y1:y2,x1:x2],
8, cv2.CV_32S)
            stats = stats[2:]
            cant_caracteres_7, cant_palabras_7 = contar_caracteres_palabras(stats,0,25)
            formulario['Comentarios'] = 'OK' if cant_caracteres_7 > 0 and cant_caracteres_7 <= 25 else 'MAL'
    return formulario
```

IMÁGENES:

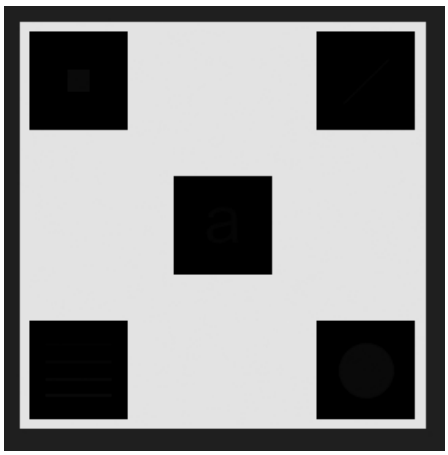


Imagen con detalles escondidos

FORMULARIO A		
Nombre y apellido		
Edad		
Mail		
Legajo		
	Si	No
Pregunta 1		
Pregunta 2		
Pregunta 3		
Comentarios		

FORMULARIO A		
Nombre y apellido	JUAN PEREZ	
Edad	45	
Mail	JUAN_PEREZ@GMAIL.COM	
Legajo	P-3205/1	
	Si	No
Pregunta 1	X	
Pregunta 2		X
Pregunta 3	X	
Comentarios	ESTE ES MI COMENTARIO.	

Formulario vacío / Formulario 1

FORMULARIO A		
Nombre y apellido	JORGE	
Edad	4500	
Mail	JORGE @GMAIL.COM	
Legajo	X45ASLAB W45	
	Si	No
Pregunta 1		
Pregunta 2	X	x
Pregunta 3		xx
Comentarios	ESTE ES UN COMENTARIO MUY MUY LARGO.	

FORMULARIO A		
Nombre y apellido	LUIS JUAN MONTU	
Edad	88	
Mail	JORGE_85@GMAIL.COM	
Legajo	M-9874/1	
	Si	No
Pregunta 1	X	
Pregunta 2	X	
Pregunta 3	x	
Comentarios	COMENTARIO DE LUIS.	

Formulario 2 / Formulario 3

FORMULARIO B		
Nombre y apellido		
Edad	1 5	
Mail	CASILLAMUYLARGAD@GMAIL.COM	
Legajo	M-98784/1	
	Si	No
Pregunta 1	O	
Pregunta 2	Oo	
Pregunta 3	ooo	
Comentarios		

FORMULARIO B		
Nombre y apellido	PEDRO JOSE GAUCHAT	
Edad	8	
Mail	PEDRO_JOSE@GMAIL.COM	
Legajo	G-6721/0	
	Si	No
Pregunta 1	SI	
Pregunta 2		NO
Pregunta 3	SI	
Comentarios	COMENTARIO DE PEDRO	

Formulario 4/ Formulario 5