

# Trabajo Integrador 2023

## Seminario de Lenguajes Opción Python

El objetivo principal de la segunda parte del trabajo integrador es completar las funcionalidades principales de la aplicación UNLPIImage y realizar análisis de datos con toda la información generada por la aplicación. Para ello, las personas responsables del proyecto deberán implementar soluciones para las dos ventanas principales de la aplicación, siguiendo los sketches proporcionados como guía.

Es importante destacar que no es necesario respetar fielmente los sketches, ya que las personas desarrolladoras pueden aplicar cambios que consideren necesarios para mejorar la solución. Sin embargo, para validar cualquier tipo de cambio, es necesario mantener una comunicación constante con el ayudante asignado para determinar si lo implementado cumple con lo pedido.

Además de completar la funcionalidad de las ventanas faltantes, se solicita que se implementen una serie de notebooks utilizando Jupyter Notebook y una aplicación usando Streamlit<sup>1</sup>(opcional), donde se realice el análisis de la información generada por la aplicación en distintas partes, como perfiles, clasificación de imágenes, eventos de la aplicación, collages y memes generados. Para ello, se deberán aplicar las distintas prácticas de análisis de datos adquiridas durante la cursada, incluyendo la limpieza y transformación de datos, el análisis exploratorio, la visualización de datos, la modelización y la evaluación de modelos. Todo este proceso debe quedar registrado y fundamentado en el notebook o aplicación generada con Streamlit.

Esta será la etapa final del proyecto integrador y, una vez finalizado, determinará la aprobación o no del proyecto.

**Notas** en base a las observaciones surgidas en la **primer entrega**:

- Usar paths relativos (NO ABSOLUTOS) en los diferentes archivos como configuración, perfiles, logs, archivos de imágenes etiquetadas y otros que se utilicen en el desarrollo del trabajo, como así también el manejo de los paths en el código.
- Todos los paths incluyendo los almacenados en archivos de configuración y los utilizados en el código deberán ser manejados de forma tal que funcionen en cualquier sistema operativo (por ejemplo tener en cuenta que los separadores de paths son diferentes entre Windows y Linux/Mac).
- utilizar el manejo de excepciones en los casos que sea necesario de forma correcta.
- entregar el trabajo con al menos:
  - 5 perfiles creados
  - 10 imágenes etiquetadas
  - 40 líneas de logs del uso del sistema

---

<sup>1</sup> La herramienta Streamlit se explicará más adelante

# UNLPImage

Continuando con el desarrollo comenzado en la entrega anterior se debe terminar la implementación de las siguientes ventanas:

- Generador de memes
- Generador de collage

Además se deberá agregar a nuestro log del sistema la posibilidad de guardar los nuevos eventos generados por las acciones de estas ventanas.

## Generador de memes

La funcionalidad "Generar meme" de **UNLPImage** debe permitir agregar textos a una imagen base para generar un meme (de forma similar a <https://imgflip.com/memegenerator>). Las imágenes base (imágenes sin texto) que se podrán utilizar como fondo para generar los memes deben ser imágenes almacenadas en el **directorio de repositorio** configurado. Este directorio puede ser modificado en la pantalla de configuraciones (implementado en la etapa 1).

Además **UNLPImage** cargará los templates para los memes desde un archivo JSON. El template de cada meme define, para cada **nombre de archivo** de la imagen: un **nombre descriptivo** del meme y una **secuencia de rangos de coordenadas**, cada rango de coordenadas determina la esquina superior izquierda y la esquina inferior derecha que delimitan un texto en el meme (tener en cuenta que un meme puede tener uno o más textos) y además tiene el nombre del archivo de la imagen base.

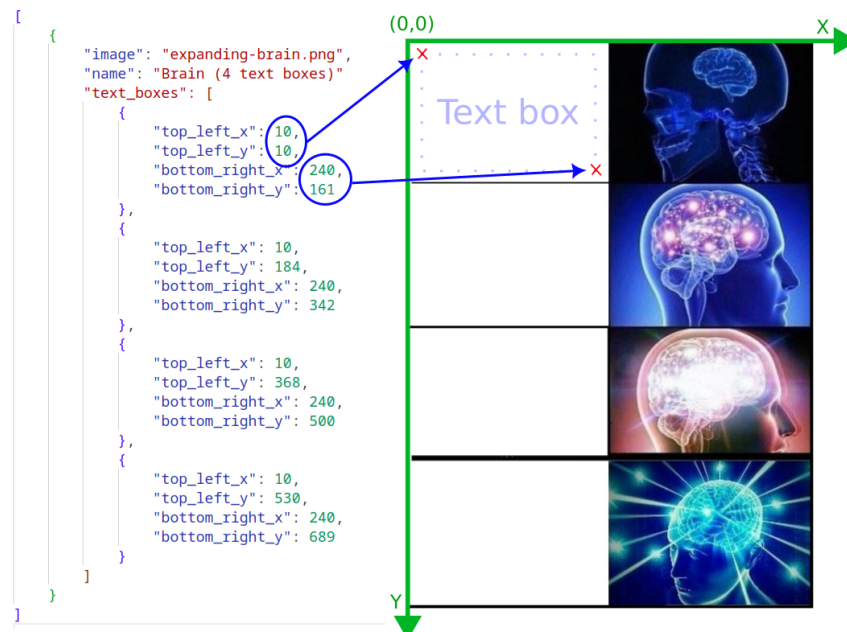


Fig.1. Ejemplo de archivo JSON de templates y la relación de sus campos con las coordenadas donde iría el texto.

Este archivo JSON debe guardarse dentro de algún directorio del proyecto.

La tipografía del texto puede ser elegida por el usuario al momento de realizar el meme. Por otro lado, el tamaño de la tipografía debe ser ajustado automáticamente por la aplicación para que el texto entre en el rango de coordenadas dado en el template correspondiente. La Fig. 3 muestra un ejemplo de cómo se podría ver la interfaz para esta ventana.

Por último esta pantalla deberá permitir guardar la imagen generada en formato JPG o PNG en el **directorio de memes** configurado. Al guardar el meme se registrará un evento con datos del meme en el log del sistema (ver "Logs del sistema").

## Notas:

- Para permitir una fácil implementación en PySimpleGUI se recomienda dividir la funcionalidad en 2 pantallas: una para seleccionar el template y otra para agregar los textos y guardar la imagen.
- El archivo JSON puede ser generado manualmente con un editor de texto. No es necesario que **UNLPImage** lo genere. Opcionalmente, el JSON se puede generar con <https://python-unlp.github.io/coordinategen/>. Opcionalmente, el JSON se puede generar con <https://python-unlp.github.io/coordinategen/>.
- El archivo JSON contendrá el nombre de archivo de la imagen (sin path), ya que la imagen se tomará del **directorio de repositorio**.
- Si una imagen no se encuentra en el archivo JSON de templates, entonces no podrá ser usada para generar un meme dado que no se tienen las especificaciones para su creación.
- Para simplificar el posicionamiento de textos usar siempre imágenes base siempre escaladas a un tamaño fijo que consideren conveniente.

## Pantalla de selección de template

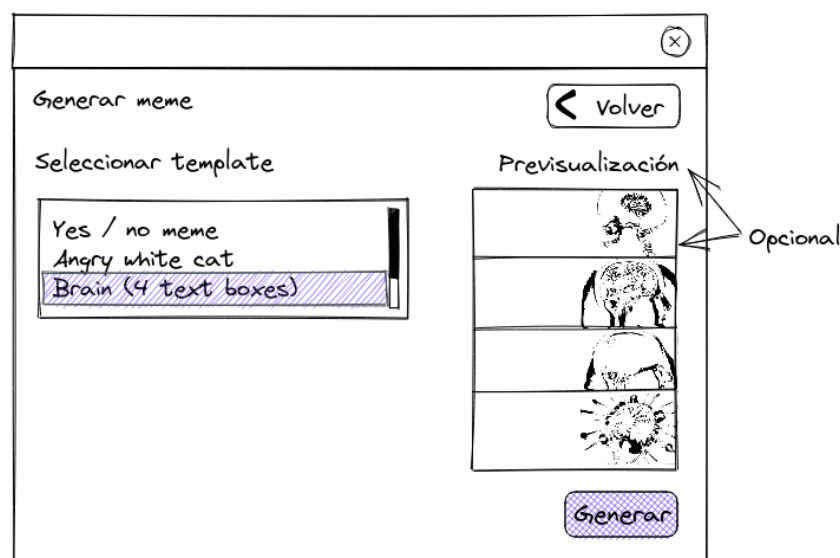


Fig. 2. Ejemplo de pantalla para seleccionar el template

## Pantalla de generación de meme

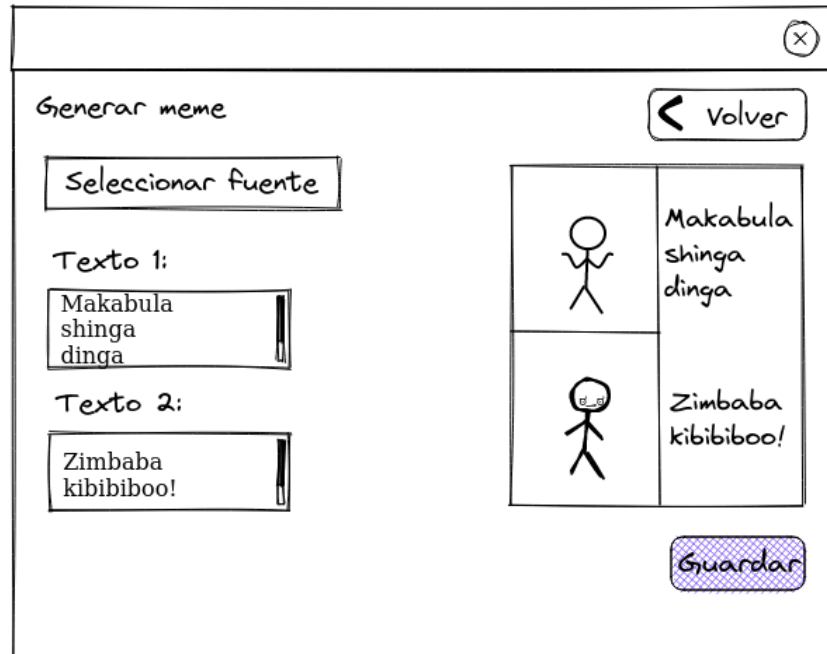


Fig. 3. Ejemplo de pantalla de generador de memes.

## Modo de uso

Para generar un meme se espera que el usuario realice los siguientes pasos:

1. Hará clic en “Generar meme” en el menú principal.
2. Seleccionará el template (meme) a generar en la pantalla de selección.
3. Hará clic en “Generar” para que se abra la pantalla de generación de memes.
4. Completará los campos de texto mostrados.
5. Hará clic en “Guardar”

## Generador de collage

La pantalla “Generar collage” de **UNLPImage** debe permitir generar un collage a partir de un “diseño de collage”, un título y una serie de imágenes previamente etiquetadas y almacenadas en el **directorio de repositorio** configurado.

Al igual que en el caso de los memes, dispondremos de varios diseños de collage que nos servirán para generar nuestros collages. Pero a diferencia de los memes, los diseños deben estar predefinidos en el código de la aplicación.

La pantalla deberá permitir:

- Seleccionar un “diseño de collage”. Debe poder elegir entre 4 diseños como mínimo.

- Seleccionar una serie de imágenes etiquetadas del repositorio de imágenes configurado. La cantidad de imágenes que se pueden seleccionar depende del “diseño de collage” seleccionado.
- Escribir un título que se agregará a la imagen en una ubicación fija (por ejemplo abajo a la izquierda).
- Guardar el collage generado en un archivo con formato JPG o PNG en la carpeta de collages seleccionada. Al guardar el collage se registrará un evento con datos del collage en el log del sistema (ver “Logs del sistema”).

## Notas:

- Para permitir una fácil implementación en **PySimpleGUI** se recomienda dividir la funcionalidad en 2 pantallas: una para seleccionar el diseño del collage y otra para agregar las imágenes, el título y guardar el collage.
- Los diseños deberán estar predefinidos en la aplicación. Un usuario no puede agregar, modificar ni borrar diseños.
- Las imágenes tendrán que ser acomodadas para que entren dentro del diseño elegido de la mejor manera.

## Pantalla de selección de diseño de collage

Los siguientes ejemplos (Fig. 4) ilustran 6 posibles diseños de collage con distintas cantidades de imágenes. Las imágenes aparecerán exactamente donde se indica en el diseño (lugar ilustrado aquí con un cuadrito), mientras que el título se ubicará en una posición fija del collage, por ejemplo en la esquina inferior izquierda.

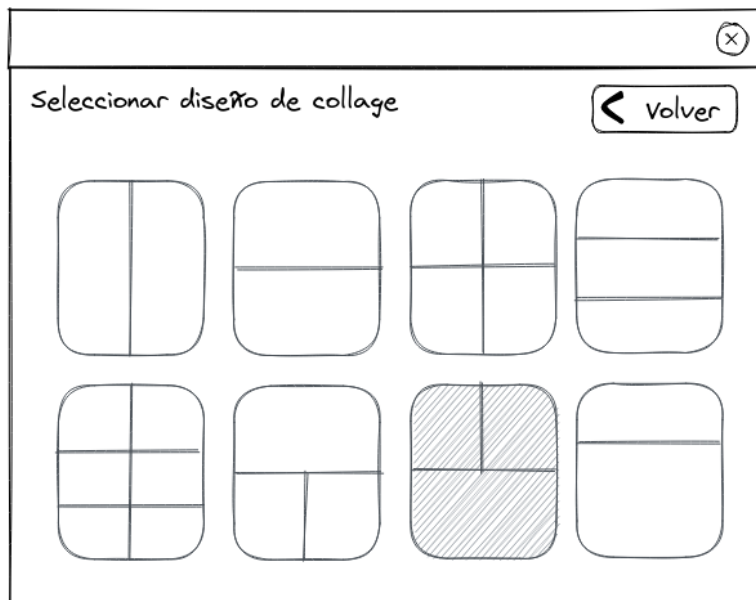


Fig. 4. Ejemplo de pantalla de selección de diseño de collage.

## Pantalla de generación de collage

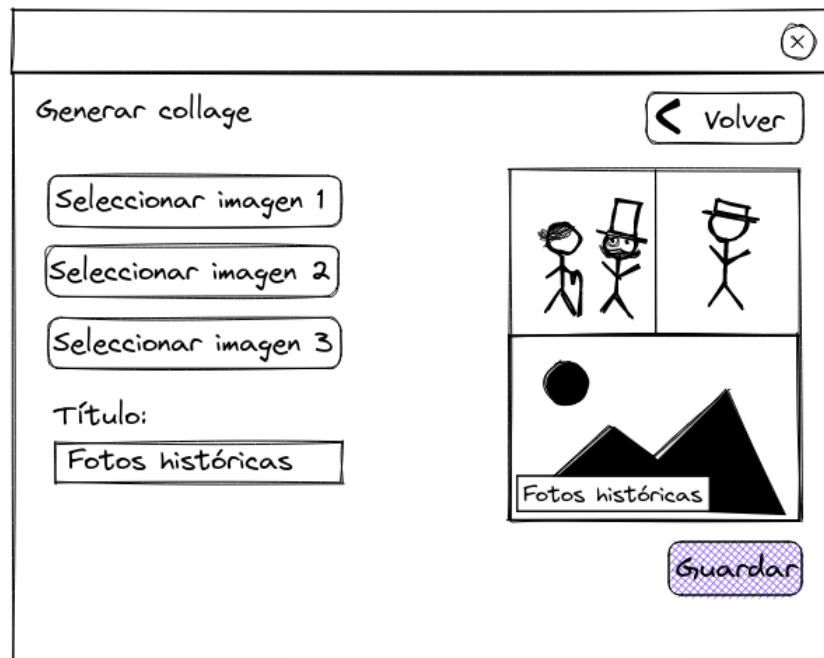


Fig. 6. Ejemplo de pantalla de generador de collages.

## Modo de uso

Para generar un collage se espera que el usuario realice los siguientes pasos:

1. Si las imágenes a utilizar no fueron etiquetadas previamente: el usuario etiquetará las imágenes. Las imágenes etiquetadas quedarán almacenadas en el repositorio de imágenes.
2. Irá a la pantalla para seleccionar un diseño de collage haciendo clic en el botón "Generar collage" del menú principal.
3. Hará clic en un diseño predefinido de collage. Esto abrirá la pantalla para generar el collage.
4. Seleccionará las imágenes que se mostrarán en los distintos cuadrantes del "diseño de collage" seleccionado.
5. Hará clic en "Guardar"

## Logs del sistema

El archivo csv de logs del sistema realizado durante la primera parte que contenía:

- fecha y hora (timestamp),
- nick o alias del perfil que realizó la acción,
- operación

Donde las operaciones solicitadas hasta ahora eran:

- cambio en la configuración del sistema,
- nueva imagen clasificada,
- modificación de imagen previamente clasificada.

A este archivo, deberán sumarse nuevas opciones a la columna de “operación” las cuales corresponden a:

1. Generación de meme
2. Generación de collage
3. Modificación de perfil
4. Creación de nuevo perfil

Y agregar dos columnas nuevas, “valores” y “textos”. Las columnas “valores” y “textos” contendrán datos según corresponda la operación que se realiza:

1. en el caso de Generación de meme se debe agregar:
  - a. en la columna “valores” la imagen que se utilizó para armar el meme
  - b. en la columna “textos” los textos agregados en el meme.
2. en el caso de “Generación de collage” se debe agregar:
  - a. en la columna “valores” los nombres de las imágenes utilizadas para generar el collage, por ejemplo, “uno.png; dos.png”.
  - b. en la columna “textos” el título del collage agregado

Ejemplo de algunas líneas del log:

```
"timestamp", "nick", "operación", "valores", "textos"
1684677032, "gali", "modificar_perfil", ,
1684677140, "gali", "nueva_imagen_clasificada", ,
1684677421, "papi", "nuevo_meme", "tres_dragones.jpg", "Linux;Mac;Windows"
1684678394, "gali", "nuevo_collage", "montañas.jpg;paisaje.jpg;río.jpg", "Vacaciones"
```

# Análisis de Datos

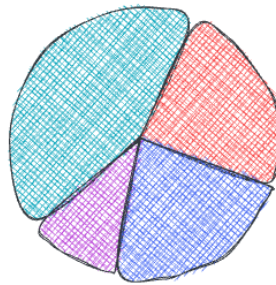
**UNLPImage** genera durante su uso dos archivos que guardan información sobre las **imágenes clasificadas** y los **logs del sistema**. De cada uno de los archivos vamos a generar estadísticas. A continuación detallaremos las estadísticas solicitadas que se calcularán utilizando la librería Pandas:

## Archivo imágenes clasificadas

En este archivo se registraron las imágenes que se fueron clasificando, con la información de cada una. De estos datos guardados en el archivo csv solicitaremos que se realicen las siguientes estadísticas:

- Gráfico de torta que muestre los porcentajes según el tipo de imagen.

Tipos de imágenes



- Calcular los valores máximos de ancho y de alto de las imágenes clasificadas.

Máximo alto: 2390px  
Máximo ancho: 3000px

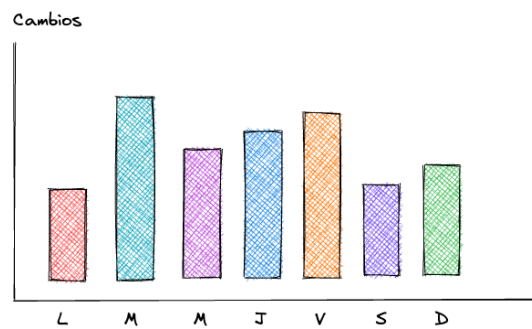
- Generar un gráfico de dispersión para visualizar la relación entre el ancho y el alto de las imágenes.

Tamaño de imágenes

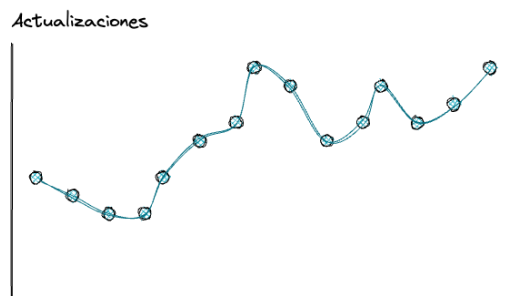




- En base a la fecha de última actualización, cantidad de cambios realizados para cada día de la semana (posible gráfico de torta o barras).



- Crear un gráfico de líneas para visualizar la evolución de la cantidad de actualizaciones a lo largo del tiempo.



- Con la lista de tags generar una nube de palabras.



- Informar cuáles son los 3 tags más utilizados.

#### Tags más utilizados

#	Tags	Usos
1	landscape	1243
2	animal	1211
3	funny	1208

- Calcular el tamaño en bytes promedio de las imágenes actualizadas **por cada perfil**, incluir los perfiles que no hayan realizado actualizaciones.

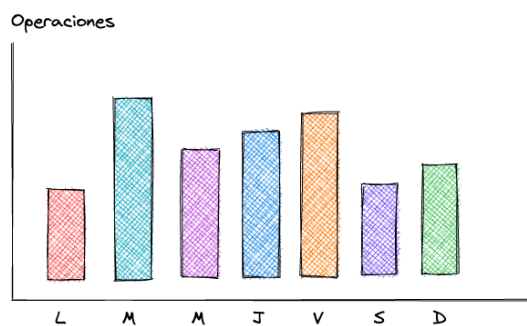
### Imágenes promedio

#	Perfil	Tamaño promedio
1	Agus	5.3MB
2	Sofi	3MB
3	Josue	7MB
4	Fer	720KB
5	Emi	1.3MB

## Archivo log del sistema

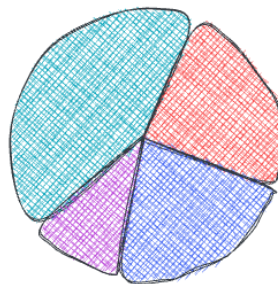
Con los datos almacenados en este archivo sobre el uso de la aplicación se deberá realizar las siguientes estadísticas utilizando Pandas:

- Realizar un gráfico comparando los días de la semana en que se realizaron operaciones usando la aplicación.

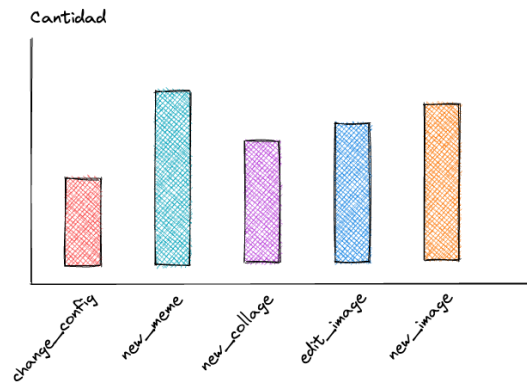


- Relacionando el archivo de logs con el archivo de perfiles generar un gráfico que muestre los porcentajes de uso de la aplicación por género.

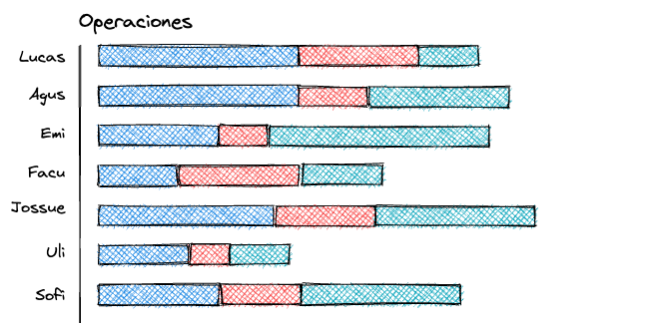
### Uso por género



- Generar un gráfico que refleje las cantidades de cada operación realizada.



- Generar un gráfico de barra apilado que muestre las cantidades de operaciones por nick.



- Generar un ranking de las 5 imágenes más usadas para generar memes y otro para generar collages.

#### Imágenes más usadas para collages

#	Imagen	Usos
1	cat.png	101
2	dog.png	93
3	mountain.jpg	80
4	sunset.jpg	50
5	lake.jpeg	49

#### Imágenes más usadas para memes

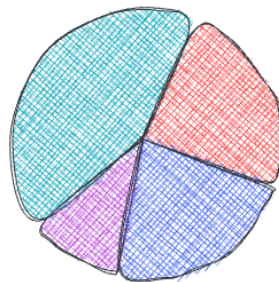
#	Imagen	Usos
1	pain_harold.png	40
2	success-kid.png	37
3	disaster-girl.jpg	31
4	bad_luck_Brian.jpg	30
5	Unimpressed Chloe.jpeg	29

- Generar una nube de palabras de los textos agregados en los collages y otra con los textos agregados en los memes



- Con los datos del archivo de perfiles generar un gráfico de torta con los porcentajes según género de las personas que realizaron las operaciones:
  - Nueva imagen clasificada.
  - Modificación de imagen previamente clasificada.

Operaciones por género



## Entrega de las estadísticas

Las estadísticas realizadas deberán entregarse en los siguientes formatos:

1. Las estadísticas correspondientes al archivo de **imágenes clasificadas** se deberán entregar en un archivo **Jupyter** donde se refleje en celdas markdown los pasos realizados para lograr las estadísticas solicitadas junto con el código correspondiente. El archivo deberá llamarse **estadisticas\_jupyter.ipynb**.
2. Las estadísticas solicitadas del archivo **logs del sistema** se deberán entregar en un archivo **Jupyter** en el mismo que utilizaron en el punto 1 o en una aplicación utilizando **Streamlit** donde se pueda visualizar las estadísticas explicando el significado del proceso realizado más los gráficos correspondientes, en este caso el archivo deberá llamarse **estadisticas\_streamlit.py**.

# Consideraciones de la entrega y defensa

Si bien el trabajo es grupal, **la defensa es INDIVIDUAL**.

La defensa se realiza dentro del horario de consulta inmediatamente posterior a la fecha de entrega. La misma consiste en tener un **encuentro con el ayudante asignado de forma virtual** donde el mismo va a realizar una serie de preguntas sobre la entrega. Estas preguntas estarán destinadas a los distintos miembros del grupo. La idea es que cada estudiante responda las preguntas destinadas a su persona teniendo un espacio individual para responder.

El desempeño en las respuestas, sumado a la participación en las consultas y el registro commits que se puede observar en GitLab serán los parámetros que el ayudante tendrá en cuenta a la hora de definir la nota de la entrega para cada integrante.

**Esto quiere decir que por más que sea una entrega grupal, las notas entre los miembros de los grupos pueden ser diferentes.**

Repetimos que es importante también tener **participación activa en las consultas prácticas y en el repositorio de GitLab** para que el ayudante tenga un panorama conceptual completo del conocimiento de cada participante del grupo.

Al trabajar con imágenes el tamaño del repositorio puede crecer considerablemente, por lo tanto le pedimos no exceder los 10MB el total del proyecto.

## Requerimientos generales

- Incluir un archivo README.md donde explique el paso a paso de como poner en marcha la aplicación. Siguiendo los pasos de esa guía el ayudante debería poder poner en marcha la aplicación desde 0 y poder probar todas las funcionalidades del sistema.
- LICENSE con la licencia del código.
- Todas las librerías junto con el número de versión utilizada en este trabajo deberán estar correctamente listadas en el archivo requirements.txt de forma que sea simple instalar todo lo necesario para que la aplicación y los notebooks funcionen.
- El sistema deberá estar preparado para ser utilizado en cualquier sistema operativo. Por ejemplo, se debe tener en cuenta que el sistema pueda ser configurado con los diferentes tipos de paths (ya sea de Windows, Linux o Mac) y funcionar correctamente para cualquier caso.
- El código debe cumplir con las guías de estilo de Python. Al menos con las reglas básicas especificadas por la cátedra. Fuentes:
  - [Documentación oficial en inglés](#)
  - [Guía básica en español de la cátedra](#)
- El código debe estar correctamente documentado utilizando docstrings. En las guías de estilo se dan ejemplos de una correcta documentación utilizando docstring.

## Fechas de entrega

La fecha de entrega tiene fecha límite el **Viernes 16 de Junio a las 23:59hs y la defensa individual se desarrollará la semana siguiente en el horario de la práctica correspondiente (Jueves 22 o Viernes 23 según corresponda).**

Dentro del portal de Cátedras encontrarán 2 tareas referidas a esta entrega.

- En una tendrán la devolución del trabajo grupal. El ayudante les dará la devolución y sugerencias sobre el código desarrollado (fecha de cierre 16 de Junio).
- La otra será para que cada participante tenga el feedback de la evaluación individual de la defensa (fecha de cierre 23 de Junio).

**Dado que el código se completa dentro del repositorio de Gitlab no hará falta que realicen ninguna entrega en estas tareas.**

## Criterios de evaluación

- Funcionalidad implementada de acuerdo al enunciado.
- Cumplimiento de las consideraciones planteadas.
- Código subido al repositorio de GitLab indicado.
- Participación durante el desarrollo del trabajo. Esto incluye tanto la asistencia a las consultas (virtual o presencial) y las contribuciones en el repositorio de GitLab.
- Desempeño en la defensa donde deberán demostrar los conocimientos utilizados para la realización del trabajo.

## Puntos de la primera entrega (en la defensa individual)

- **Puntos de cursada:** 40 puntos.

## Recomendaciones sobre el trabajo en grupo

- **Establecer un plan de trabajo:** es importante que los estudiantes establezcan un plan de trabajo detallado y que definan claramente las tareas que cada uno va a realizar. Esto les permitirá trabajar de manera más eficiente y organizada.
- **Releer el enunciado regularmente** para recordar las especificaciones y evitar confusiones.
- **Utilizar herramientas de comunicación:** los estudiantes pueden utilizar herramientas de comunicación como Discord para mantenerse en contacto y coordinar el trabajo. Es importante que todos estén en la misma página y se comuniquen con regularidad.
- **Utilizar un flujo de trabajo en Git:** los estudiantes deben familiarizarse con un flujo de trabajo en Git. Es importante que los estudiantes se comuniquen y se pongan de acuerdo en el flujo de trabajo que van a utilizar.

- **Establecer reuniones regulares:** los estudiantes pueden establecer reuniones regulares para discutir el progreso del proyecto y resolver cualquier problema que haya surgido. Esto les permitirá mantenerse al día y abordar los problemas de manera oportuna.
- **Utilizar comentarios en el código:** es importante que los estudiantes utilicen comentarios en el código para explicar qué hace cada sección y cómo funciona el programa. Esto facilitará la comprensión del código para los demás integrantes del grupo.
- **Fomentar el trabajo en equipo:** los estudiantes deben fomentar el trabajo en equipo y ayudarse mutuamente. Es importante que estén dispuestos a dar y recibir retroalimentación constructiva.