Resumen de Python:

- Pantalla Start:
  - Importaciones:
    Importamos de start functions: read_users, button_imgs, start, profile_modifications.
    De main importamos: main_window.
    De new_profile importamos new_profile_window.
  - En orden de aparición:

    read_users: Intenta abrir un archivo json, de no encontrarlo imprime que no lo encontró, si lo encuentra retorna una lista de diccionarios. (Debería retornar una lista vacía en caso de no encontrar el json, así nos ahorramos un try afuera).

    button_imgs: se encarga de manejar el puntero de que imagen actualizar cuando queremos mostrar los perfiles, si el puntero es positivo, entonces directamente va actualizando (un máximo de 4 por llamada) y le resta 1, si es negativo y la cantidad de usuarios en menor 3, lo resetea y sale del for (porque ya no quedan más usuarios para mostrar), sino resetea el puntero y muestra las imágenes que le faltaban.

    set_imgs: es llamado por button_imgs para actualizar una imagen de perfil, intenta agarrar de users el avatar que le toca para actualizar, de no existir tal camino muestra la imagen default, además, actualiza una lista de usuarios actuales en función del puntero, y lo retorna -1.

    profile_modifications: Actualiza el usuario elegido cuando volvemos del programa, y actualiza la imagen a mostrar. Para ello lee de disco.
  - Esta ventana va hacia la new_profile.
- Pantalla main:
  - Importaciones:
    Importa las constantes.
    Importa las distintas ventas a acceder (todas menos el start).
    Importas log.
    Load de main_functions para cargar la imagen del avatar.
  - En orden de aparición:

    menu: Crea el layout inicial de la ventana, sin cargar la imagen, usando el diccionario (debería solo pasarle el nombre ya que solo usa eso).

    En el run: asignamos el nombre global para el objeto, y intenta abrir los logs y si no existe lo crea. También en su loop llama las invocaciones de todas sus ventanas.

    load: recarga la imagen a mostrar (del usuario actual), tiene un if para evaluar si la imagen existe, ya que con excepciones no funciona porque la levanta el pysimplegui antes de poder manejarla nosotros, si no existe muestra el default.

    - Esta ventana va hacia todas las ventas.
- Help:
    - Lo más raro a saber es lo de windows.py que tiene un sistema de tabs que usa una lista de listas mostrar las diferentes pestañas de datos.
    - Vuelve a main.

- New profile:
    - Importaciones: se invoca las constantes, sus funciones y la main_window.
    - Va a main.
    - En la creación de los elementos de UI usa un input vacío e invisible a lado del filebrowse para no guardar en el combo lo que te llega.
    - Cuando vuelve al de Tomi si no creamos usuario usa delete_img_before_back que elimina la imagen elegida si se eligió una (volver al de Tomi conlleva cancelar la creación de usuario).
    - Funciones:
        - Clear_inputs: Te pone en blanco las casillas de inputs en caso de escribirlas por primera vez o borrar cuando te tira un error (Se puede modularizar haciendo que reciba el input que disparo el evento, en vez de la ventana entera, entonces no usarías 3 ifs por invocación).
        - New_user: usa read_inputs_new para leer los datos puestos en las casillas de la ventana, chequea si el usuario existe con user_existe, y lo valida con validate_user e inputs. Además, si está bien validado lo agrega con new_json_user.
        - Read_inputs_new: lee de los inputs, chequea que el checkbox este activado para levantar el input ocultado, formatea el camino de la imagen, guarda la imagen redondeada en base a la elegida, y por fin retorna un diccionario con los datos del usuario. (Se debería separar el guardado de la imagen en otra función).
        - User_exist: Chequea si el usuario existe en el json o si el archivo de usuarios existe. (No se debería guardar la constante en una variable si después no la modificamos, asignar JSON a data no es necesario, directamente usar JSON, además retorna none (cuando no existe el usuario en el listado)).
        - Validate_user: chequea que los datos estén entre los parámetros deseados (edad < 75, etc), te retorna true o false.
        - New_json_user: Abre el archivo de usuarios, si ya existe lo lee y en un variable pone sus datos y le agrega los del nuevo usuario, sino pone en una lista nueva la data del usuario, y después lo guarda sí.
        - Validate_inputs: Te colorea las casillas del UI en base a que no está entre los parámetros elegidos, hace la diferenciación de chequear el Nick o no en base si es solo para leer (modificar el usuario) o no [Tal vez convendría juntarlo con validate user].
    - Notas: Cuando guardamos el usuario lo guardamos y después lo leemos devuelta, conviene guardarlo la primera vez que lo leemos para así reutilizarlo.
- Edit_profile:
    - Recibe el nombre desde la pantalla de feli, y con eso busca los datos del usuario para plasmarlo en las casillas (además guarda se guarda los datos del archivo en un

diccionario), cuando le da a guardar se lo guarda haciendo las validaciones necesarias para ver si está bien lo que escribió el usuario.
(Acá debería directamente recibir el usuario actual de feli, porque sino estas leyendo 2 veces de un archivo, cuando directamente podrías escribirlo de 1).
(No cierra le ventana nunca, porque hace return y la esconde, directamente habría que hacer un break y retornar después de hacer window.close()).

(Ya que usa el mismo layout que el otro, eso se podría modularizar).

Labelling:

Importaciones: algo

Vuelve a la main.

Funciones:

Show_image: recibe el nombre de la imagen, el directorio donde están las imágenes y la ventana. Verifica si existe la imagen y el archivo en metadata, si existe la imagen la muestra y actualiza la ventana con las etiquetas que están en el archivo de metadata. Si no hay archivo de metadata lo crea. Si guardas, guardas los datos.

Update_csv: recibe la imagen y los tags, la descripción,

Edit_img_csv: recibe datos a escribir en el csv. Lo busca y escribe los datos en el. Luego de escribir lo escribe en el archivo de log.

Settings:

Importaciones: algo

Vuelve a la main.

Busca en el json

Save_config: creo o sobrescribe el json con los datos.

Escribe lo que hace en el log

Reset devuelve los caminos a su default

Meme

No

Collage:

No

Log:

Clase con métodos

Try_open_logs: cheque que exista el archivo logs.

Write_logs: guarda los datos en el log.