Informe Final
Evaluación de accesibilidad de la aplicación UNLPImage

Pedro G. Villegas – *pedrogvillegas@gmail.com*

Seminario de Lenguajes: Python (2023)
Facultad de Cs. Astronómicas y Geofísicas – UNLP
Profesora: Claudia Banchoff
Facultad de informática – UNLP

Última modificación: 14 de agosto de 2023

Índice

Introducción

En la era de la información, nos puede resultar cada vez más notoria la tendencia a comunicarnos por medio de contenido audiovisual. Redes sociales, páginas web y hasta configuraciones predeterminadas en los dispositivos buscan poseer diseños originales, que –al menos en la forma occidental de hacer las cosas– intentan permanecen simples y minimalistas, pero están cargados de animaciones y formas de interactuar con el contenido. Tomemos por ejemplo el diseño web de la página principal de Apple o Tesla. Notemos como intentan acceder a nosotros por medio de colores llamativos, animaciones, desplazamientos de plantillas y formas innovadoras de interactuar, herramientas que utilizan para que navegar dentro de sus servidores sea más que placentero para sus clientes y así destacar sus sitios respecto del resto de internet.

Pero, en el proceso de diseñar e innovar, es lógico pensar que en herramientas digitales el diseño y la estética pasen principalmente por los ojos. Este es el punto en el que nos vamos a centrar, pues, según la Organización Mundial de la Salud (OMS), *"En todo el mundo, al menos 2.200 millones de personas tienen deficiencia visual"*[1], y es en este tipo de deficiencias que nos podemos encontrar con discapacidades como la ceguera o el daltonismo. Cuando el diseño no tiene en cuenta estas variables, hay sectores de la población que quedan privados de acceder a herramientas, productos y servicios, y nosotros como programadores y diseñadores de los mismos podemos evitar caer en este conflicto.

En este informe realizaremos una evaluación de accesibilidad de la aplicación UNLPImage desarrollada por Celi Marcos, Curin Daniela y Villegas Pedro, durante la cursada del año 2023 del Seminario de Lenguajes de Python. En el mismo, analizaremos distintos aspectos de la aplicación desde un punto de vista que no ha sido tenido en cuenta durante el desarrollo de la misma, y veremos cuán distinta es la experiencia de ejecutar nuestra aplicación por una persona daltónica o no vidente.

[1] https://apps.who.int/iris/bitstream/handle/10665/331423/9789240000346-spa.pdf, página 26.

1. Aspectos de accesibilidad analizados (cambiar nombre)

Como hemos anticipado, en este informe analizaremos la experiencia de ejecución de nuestro programa por una persona daltónica y una no vidente.

El daltonismo es *"una alteración de origen genético que afecta a la capacidad de distinguir los colores."*[2]. A pesar de que existen muchos tipos de daltonismo, el 99 % de los casos corresponden a los tipos dicromáticos, en los que la persona afectada no posee uno de los tres tipos de fotorreceptores oculares, o estos son disfuncionales. Así, los daltonismos más típicos son la **deuteranopía** y la **protanopia**, que perciben la gamma de colores del espectro visible como podemos observar en la figura 1, afectando alrededor del ~ 8 % de la población masculina y al ~ 0.5 % de la población mundial femenina. A estos tipos de daltonismo también se los llama del tipo **rojo-verde**, pues son los principales colores indistinguibles para quienes los padecen.

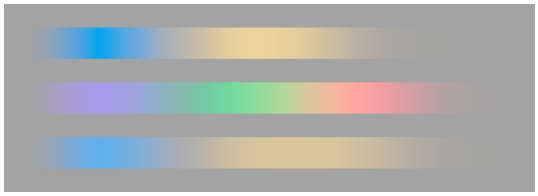

Figura 1: *En el centro, espectro visible de colores visto por una persona con visión normal. Arriba, el mismo espectro visto por una persona que padece protanopia, abajo por una persona con deuteranopía. Fuente: Wikimedia Commons.*

Por otro lado, la ceguera es una discapacidad que afecta a una amplia porción de la población mundial. Según la OMS, *"Aunque se desconoce el número exacto, se estima que 11,9 millones de personas en todo el mundo tienen una deficiencia visual moderada o grave o ceguera que podría haberse evitado, a causa del glaucoma, la retinopatía diabética y el tracoma."*[3].

Para reducir la discriminación y fomentar la inclusión, es importante contemplar la posibilidad de tener un público que padezca una deficiencia visual a la hora de desarrollar software, y hacer un esfuerzo en adaptar nuestras tecnologías para que sean accesibles para esta y otras minorías.

2. Herramientas utilizadas

Puesto que la ceguera y el daltonismo son discapacidades que afectan de forma diferente la visión, utilizaremos distintas herramientas para evaluar nuestra aplicación: el simulador de ceguera de color de Colblindor, Coblis; y el lector de pantalla integrado de Windows 10, Narrador.

2.1. Coblis: simulador de ceguera de color

El funcionamiento de Coblis es muy sencillo. Precisa que carguemos imágenes propias para poder evaluar cómo vería una persona que sufre de daltonismo la misma imagen. En nuestro caso, aprovecharemos esta herramienta para evaluar si la paleta de colores de nuestra aplicación es apropiada para personas que sufren de daltonismos del tipo rojo-verde.

2.2. Narrador de Windows: lector integrado de pantalla

El Narrador de Windows es una herramienta más compleja que la anterior puesto que está diseñada para que personas no videntes puedan utilizar computadoras sin tener que recurrir al mouse o touchpad. Funciona a través de atajos del teclado, como son:

- **Windows + Ctrl + Enter:** iniciar/detener el narrador.
- **Tab/flechas de dirección:** desplazarse sobre los distintos elementos en pantalla.
- **Enter:** Seleccionar elemento.
- **Bloq Mayús/Insert:** Teclas «narrador». En ellas se centra la principal funcionalidad de la herramienta.
- **Bloq Mayús + barra espaciadora:** iniciar/detener el modo examen.
- **Bloq Mayús + Ctrl + D:** Describe imágenes e hipervínculos.
- **Bloq Mayús + F5 o F6:** Navegación a través de títulos o puntos de referencia de la página.

Por supuesto que esta herramienta tiene más comandos y funcionalidades que las mencionadas, pero haremos uso principalmente de estas para evaluar su funcionamiento en nuestra aplicación.

[2] https://es.wikipedia.org/wiki/Daltonismo
[3] https://apps.who.int/iris/bitstream/handle/10665/331423/9789240000346-spa.pdf, página xi

3. Resultados

3.1. Paleta de colores

Durante el desarrollo de nuestra aplicación hemos decidido utilizar la paleta de colores `'light green 3'` del módulo `PySimpleGUI`, que podemos observar en las capturas de pantalla de la figura 2. El mismo consta de dos tonalidades de verde, diferenciando los botones del fondo, con un color de fuente negro sobre el texto plano y blanco sobre los botones. A demás de lo mencionado, el único color adicional que hemos elegido es el de los diseños de los templates de collages, que hemos optado por una tonalidad naranja para contrastar con el fondo de la aplicación.

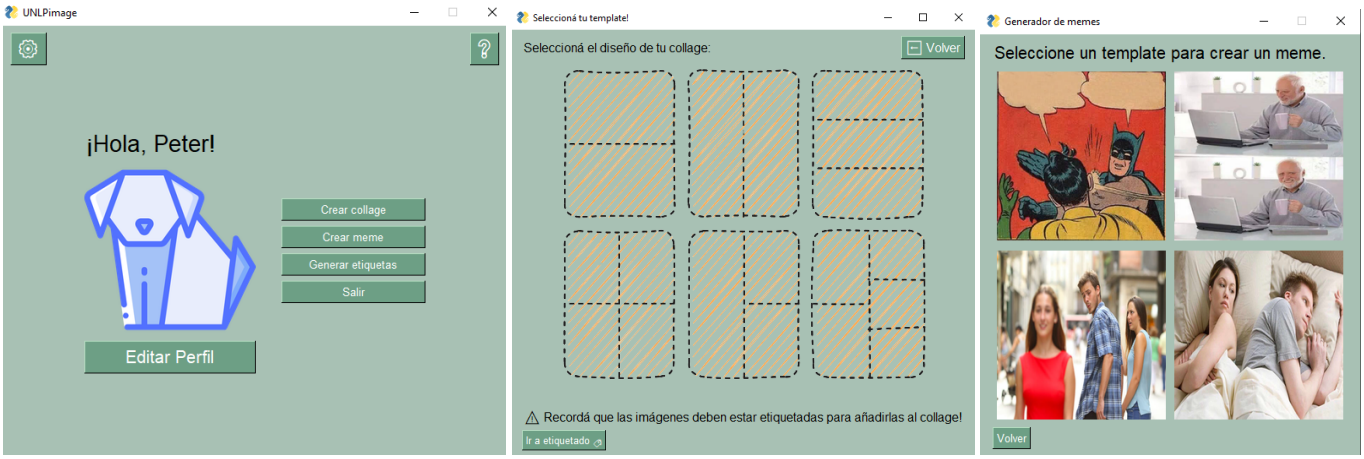

Figura 2: *Capturas de pantalla de la aplicación UNLPImage. A izquierda el menú principal, en el centro la selección de templates para la generación de un collage, y a derecha la selección de templates para realizar un meme.*

Como vimos en la sección anterior, los tipos más comunes de daltonismo son aquellos en los que las tonalidades de rojo y verde se vuelven indistinguibles, la deuteranopía y la protanopia. Haciendo uso de Coblis, hemos podido simular cómo vería nuestra aplicación una persona que padece estos tipos de daltonismo. Podemos visualizarlo en las figuras 3 y 4. Observemos como en ambos casos el contraste entre el relleno de los templates de collages con el fondo de nuestra aplicación es apenas distinguible. El texto es perfectamente legible de todas formas, y los botones son fácilmente distinguibles. Podríamos decir que, a grandes razgos, la ejecución de la aplicación para una persona daltónica no es una experiencia alejada de la que experimentaría una persona sin esta afección.

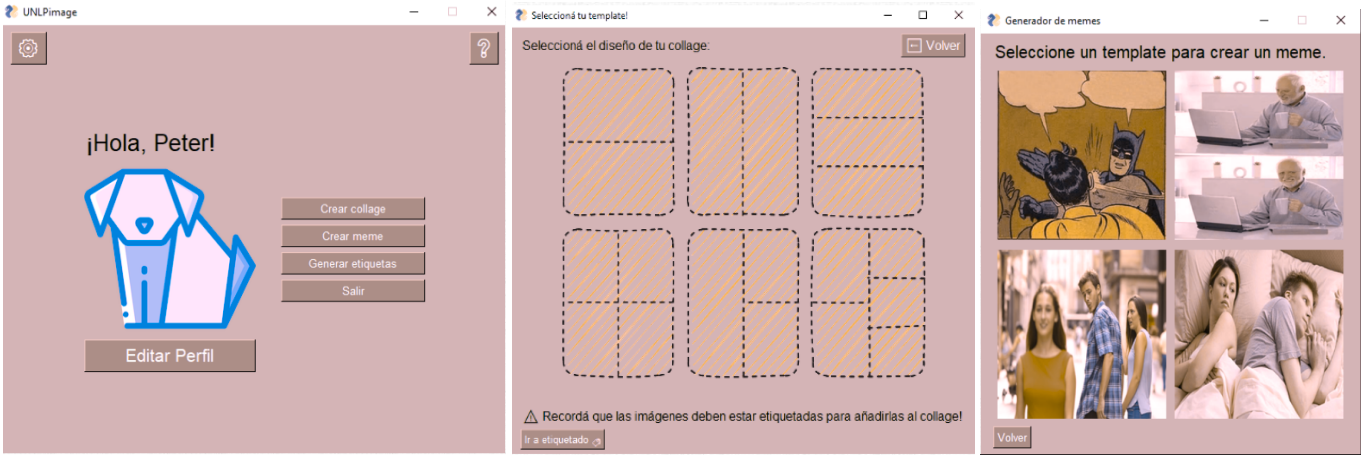

Figura 3: *Simulaciones de deuteranopia sobre las capturas de la aplicación UNLPImage.*

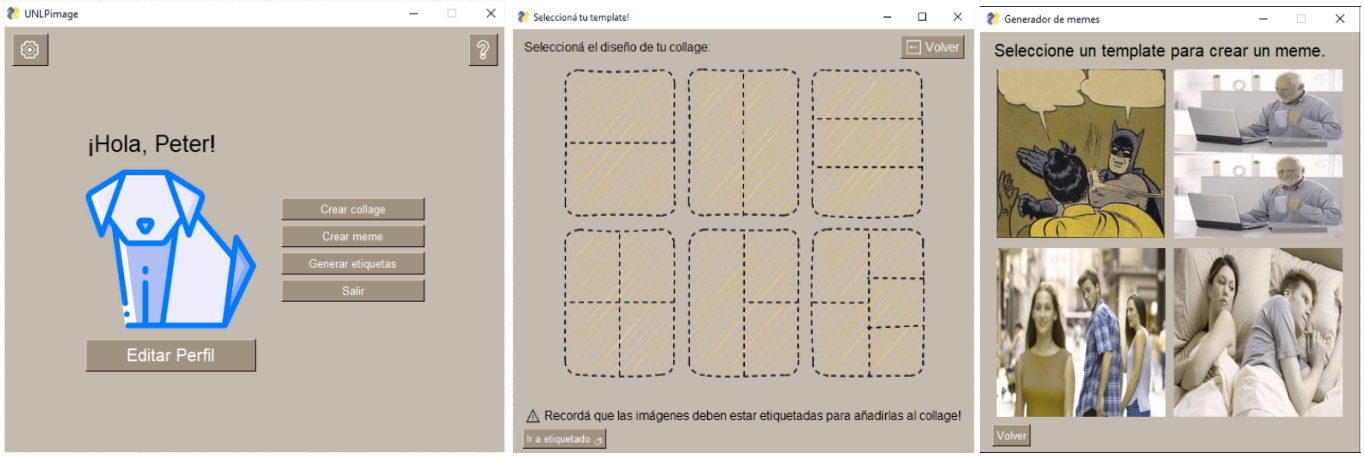

Figura 4: *Simulaciones de protanopia sobre las capturas de la aplicación UNLPImage.*

3.2. Lector de pantalla en UNLPImage

La experiencia con el narrador integrado de Windows, a diferencia de la evaluación de la paleta de colores, no fue positiva. `PySimpleGUI` según su documentación está diseñado en base a cuatro «frameworks», o Kits de herramientas de widgets: `Tkinter`, `PyQT`, `wxPython` y `Remi`. Aparentemente es problema de estos frameworks y, por tanto, del diseño general de `PySimpleGUI` que los lectores de pantalla no puedan acceder a la información presente en las ventanas desarrolladas en estos entornos.

Observemos la figura 5. En la ejecución simultánea del menú principal junto con el narrador, podemos observar que este último es capaz de reconocer los elementos de la ventana, como son los botones. Además, en esta ventana logra reconocer las imágenes, que al ejecutar el comando **Bloq Mayús + Ctrl + D** las envía a el servidor de Microsoft para identificar y luego describir (aunque, sólo en inglés) el contenido de la misma. Por ejemplo, situándonos sobre la imagen de perfil del usuario "Peter", el narrador dice: *"Obteniendo leyenda de imagen: AI can"*. El problema surge con que, dentro de la ventana, existen otros elementos que el narrador reconoce como imágenes, y esto puede resultar molesto para el usuario pues el narrador menciona su existencia en la ventana como "imágenes" cuando en realidad la funcionalidad de estos objetos es otra. En este ejemplo son el texto "¡Hola, Peter!" y los elementos de `PySimpleGUI` `sg.Push()` y `sg.Vpush()` que rellenan el espacio entre elementos de forma horizontal y vertical, respectivamente. Esto también puede verse en la figura 6.

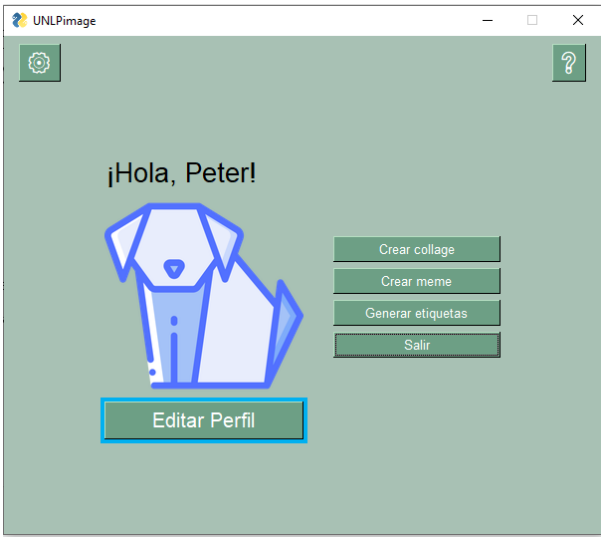

Figura 5: *Menú principal de la aplicación UNLPImage ejecutada junto al narrador de Windows, posicionado sobre el botón "Editar Perfil".*

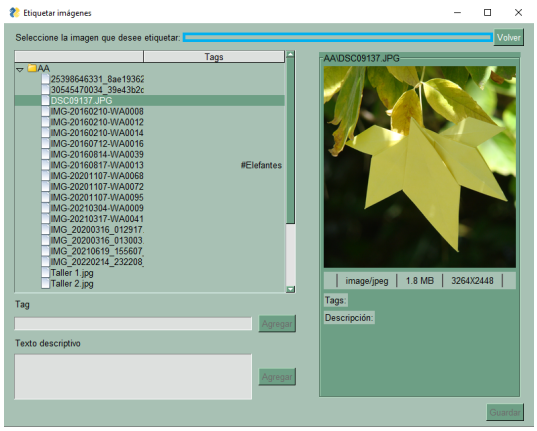

Figura 6: *Ventana de etiquetado de imágenes ejecutada junto al narrador de Windows, posicionado sobre un elemento* `sg.Push()` *de* `PySimpleGUI`.

En todos estos elementos, cuando te situás sobre ellos con el narrador, este dice *"imagen"*. El problema no termina aquí, pues, a pesar de reconocer los botones como tales, al seleccionarlos sólo nos dice *"botón"*. Al parecer, modificando la configuración del narrador tampoco es posible acceder al texto dentro de los botones. Por lo que un usuario no vidente reconocería esta ventana como una lista de *"botones"* e *"imágenes"*. Otro problema que surge al hacer este experimento, es que es frecuente que el narrador no esté situado sobre el elemento seleccionado en la ventana. Prestando atención a la figura 5, podemos notar como el narrador está posicionado sobre "Editar Perfil", mientras que el botón seleccionado es el de "Salir". Presionando la barra espaciadora en este caso nos dejaría dos posibilidades, dependiendo del modo activo del narrador: la ventana se cierra, o no hace nada.

Finalmente, hay algunos elementos de `PySimpleGUI` que el narrador no es capaz de reconocer. Por ejemplo, en la figura 6 los elementos `sg.input()`, utilizado dos veces para ingresar texto, y `sg.Tree()`, que nos permite visualizar la carpeta de imágenes y su contenido,

ambos situados en la columna izquierda de la ventana, son salteados por el narrador al recorrer cada elemento con las flechas de dirección del teclado. Esto hace que esta ventana en particular quede completamente inutilizable.

Estos hechos han sido mencionados por usuarios en foros web. Por ejemplo, en esta discusión en StackOverflow de febrero de 2022, un usuario no vidente informa que en el programa que está escribiendo no es capaz de hacer que su lector de pantalla, NVDA, reconozca el texto ingresado en elementos `sg.input()`. El mismo problema fue mencionado por otro usuario en esta consulta de febrero de 2020 en el foro oficial de GitHub de `PySimpleGUI`, donde un moderador le proporciona una posible solución añadiendo la línea de código: `window['-IN-'].Widget.accessibleName()`, donde con `Window['-IN-']` se hace referencia al objeto `sg.Input()`. También en GitHub, en las consultas #2637 y #4404 de febrero de 2020 y junio de 2021 respectivamente, de nuevo usuarios no videntes mencionan la imposibilidad de leer las ventanas generadas con `PySimpleGUI` con sus lectores de pantalla, con la diferencia que en este caso los moderadores mencionan que se está haciendo un esfuerzo para migrar el módulo a PySimpleGUIWx, donde las posibilidades de aplicaciones con más accesibilidad para usuarios no videntes son mayores. Lamentablemente, como se menciona en el último caso, esta no es una de las mayores prioridades del equipo de desarrollo del módulo:

" It's unfortunate that tkinter doesn't yet support screen readers. Your best bet is to go with WxPython for screenreader support. One of the purposes of the PySimpleGUIWx port was to get this feature into the PySimpleGUI universe. However, the PySimpleGUIWx port is not yet "feature complete" and is down the priority list a ways at the moment. Wish I had better news, but that's the honest assessment of the situation. You're not the first person that's mentioned this problem." - Moderador de `PySimpleGUI`. (Ver el mensaje en su contexto original)

4. Conclusiones

`PySimpleGUI` parece no ser un módulo apropiado para desarrollar aplicaciones de escritorio con herramientas de accesibilidad involucradas. Como parte del equipo de desarrollo de la aplicación, hemos tenido la suerte de elegir una paleta de colores que se adapta sin mayores inconvenientes para usuarios que padecen de daltonismo, pero lamentablemente nuestra aplicación es completamente inerte para usuarios que padezcan grados parciales o totales de ceguera. Para desarrollar aplicaciones que brinden esta posibilidad, actualmente es importante evaluar el entorno y las herramientas de programación de las que se harán uso. Afortunadamente, la informática, la tecnología y las herramientas de accesibilidad son entornos vivos, en los que día a día se crean dispositivos y software que facilitan la experiencia de usuarios con discapacidades. Creemos que es importante asegurarse de formar profesionales que sean conscientes de que no todos los seres humanos experimentamos la vida de la misma forma, y que existe una diversidad en estas experiencias particulares que hacen que necesitemos que las herramientas se adapten a los usuarios. De esta forma podremos asegurarnos de que, con el tiempo y muchas personas trabajando en ello, esta brecha se reducirá.

5. Referencias

- Código fuente de la aplicación UNLPImage:
  https://gitlab.catedras.linti.unlp.edu.ar/python2023/code/grupo35
- Foro oficial de `PySimpleGUI` en GitHub:
  https://github.com/PySimpleGUI/PySimpleGUI/issues
- Nota de la OMS sobre ceguera y discapacidad visual:
  https://www.who.int/es/news-room/fact-sheets/detail/blindness-and-visual-impairment
- Informe mundial sobre la visión:
  https://apps.who.int/iris/bitstream/handle/10665/331423/9789240000346-spa.pdf
- Guía del Narrador de Windows:
  https://support.microsoft.com/es-es/windows/guía-completa-del-narrador
- Más sobre el daltonismo:
  https://es.wikipedia.org/wiki/Daltonismo