ITERACIÓN 1:

```
import torch
from diffusers import StableDiffusionPipeline
print(torch.__version__)
print(torch.version.cuda)
device = "cpu" #Usar cpu, no probé con ROCm para AMD
print(f"Usando dispositivo: {device}")
model_id = "CompVis/stable-diffusion-v1-4"
pipe = StableDiffusionPipeline.from_pretrained(model_id)
pipe = pipe.to(device)
#Contador de imágenes generadas
img = 1
while True:
    prompt = input("Ingrese el prompt (dejar vacío para
terminar): ")
    if len(prompt) == 0:
        break
    image = pipe(prompt, height=512, width=512).images[0]
#Ejecutar el pipeline sin autocast
    image.save(f"img{img}.png")
    print(f"Imagen {img} guardada como img{img}.png")
    img += 1
```

Prompt: “Hamburguesa”
Salida:

pipe = StableDiffusionPipeline.from_pretrained(model_id)

model_id = "stabilityai/stable-diffusion-2-1"
model_id = "prompthero/openjourney-v4"

Res 1024x1024

Res 256x256

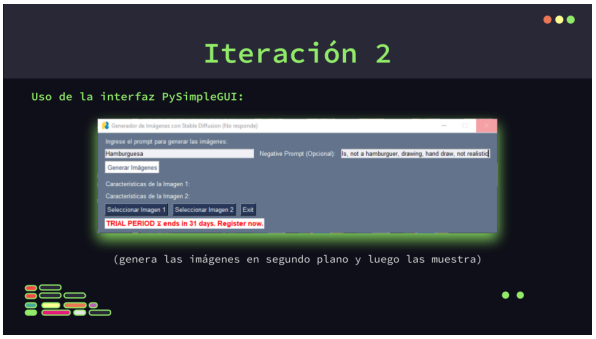
Iteración 2:

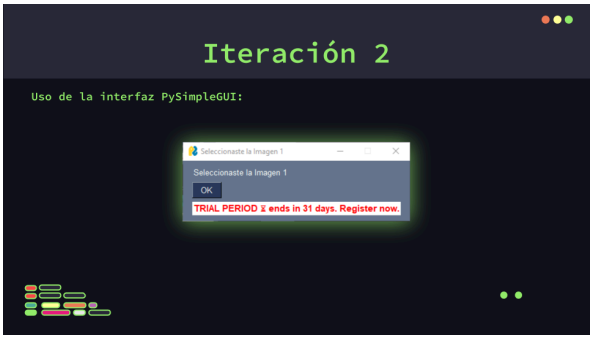
```
import torch
from diffusers import StableDiffusionPipeline
import PySimpleGUI as sg
from PIL import Image, ImageStat
import os

# Función para calcular características de la imagen
def calcular_caracteristicas(img):
    # Convertir imagen a RGB y obtener sus estadísticas
    stat = ImageStat.Stat(img)
```

```
    # Obtener promedio RGB
    rgb_avg = stat.mean[:3]  # Los tres primeros valores son R, G, B

    # Calcular brillo promedio (la media de los valores de luminancia)
    brillo_promedio = sum(stat.mean) / len(stat.mean)

    # Calcular contraste promedio
    contraste_promedio = sum(stat.stddev) / len(stat.stddev)

    return rgb_avg, brillo_promedio, contraste_promedio

# Mostrar las características en la ventana
def mostrar_caracteristicas(imagen_numero, rgb_avg, brillo_promedio,
contraste_promedio):
    return f"Imagen {imagen_numero}\nPromedio RGB: {rgb_avg}\nBrillo
promedio: {brillo_promedio:.2f}\nContraste promedio:
{contraste_promedio:.2f}"

# Cargar el modelo de stable diffusion desde Hugging Face
device = "cpu"  # Usar CPU, no se probó con ROCm para AMD
print(f"Usando dispositivo: {device}")
model_id = "stabilityai/stable-diffusion-2-1"
pipe = StableDiffusionPipeline.from_pretrained(model_id)
pipe = pipe.to(device)

# Interfaz gráfica PySimpleGUI
layout = [
    [sg.Text("Ingrese el prompt para generar las imágenes:")],
    [sg.InputText(key="prompt"), sg.Button("Generar Imágenes")],
    [sg.Image(key="img1"), sg.Image(key="img2")],
    [sg.Text("Características de la Imagen 1:", key="text1")],
    [sg.Text("Características de la Imagen 2:", key="text2")],
    [sg.Button("Seleccionar Imagen 1"), sg.Button("Seleccionar Imagen
2"), sg.Exit()]
]

# Crear la ventana
window = sg.Window("Generador de Imágenes con Stable Diffusion",
layout)

# Contador de imágenes generadas
img_counter = 1
```

```
# Bucle principal de la interfaz
while True:
    event, values = window.read()
    if event == sg.WINDOW_CLOSED or event == "Exit":
        break
    if event == "Generar Imágenes":
        prompt = values["prompt"]
        if len(prompt) == 0:
            sg.popup("Por favor, ingresa un prompt.")
            continue

        # Generar dos imágenes con el prompt dado
        img1 = pipe(prompt, height=1024, width=1024).images[0]
        img2 = pipe(prompt, height=1024, width=1024).images[0]

        # Guardar temporalmente las imágenes
        img1.save(f"img{img_counter}_1.png")
        img2.save(f"img{img_counter}_2.png")

        # Calcular características de las imágenes
        rgb_avg1, brillo_prom1, contraste_prom1 =
calcular_caracteristicas(img1)
        rgb_avg2, brillo_prom2, contraste_prom2 =
calcular_caracteristicas(img2)

        # Actualizar la interfaz con las imágenes y sus características
        window["img1"].update(filename=f"img{img_counter}_1.png")
        window["img2"].update(filename=f"img{img_counter}_2.png")

        # Actualizar textos con las características
        window["text1"].update(mostrar_caracteristicas(1, rgb_avg1,
brillo_prom1, contraste_prom1))
        window["text2"].update(mostrar_caracteristicas(2, rgb_avg2,
brillo_prom2, contraste_prom2))

    if event == "Seleccionar Imagen 1":
        sg.popup("Seleccionaste la Imagen 1")
    elif event == "Seleccionar Imagen 2":
        sg.popup("Seleccionaste la Imagen 2")

# Cerrar la ventana
window.close()
```

```
# Eliminar archivos temporales al cerrar
for file in os.listdir():
    if file.endswith(".png"):
        os.remove(file)
```





Iteración 3:

```
import torch
from diffusers import StableDiffusionPipeline
import PySimpleGUI as sg
from PIL import Image, ImageStat
import os
def calcular_caracteristicas(img):
    stat = ImageStat.Stat(img)
    rgb_avg = stat.mean[:3]
    brillo_promedio = sum(stat.mean) / len(stat.mean)
    contraste_promedio = sum(stat.stddev) / len(stat.stddev)
    return rgb_avg, brillo_promedio, contraste_promedio
```

```
def mostrar_caracteristicas(imagen_numero, rgb_avg, brillo_promedio,
contraste_promedio):
    return f"Imagen {imagen_numero}\nPromedio RGB: {rgb_avg}\nBrillo
promedio: {brillo_promedio:.2f}\nContraste promedio:
{contraste_promedio:.2f}"
device = "cpu"
print(f"Usando dispositivo: {device}")
model_id = "prompthero/openjourney-v4"
pipe = StableDiffusionPipeline.from_pretrained(model_id)
pipe = pipe.to(device)
layout = [
    [sg.Text("Ingrese el prompt para generar las imágenes:")],
    [sg.InputText(key="prompt"), sg.Text("Negative Prompt
(Opcional):"), sg.InputText(key="negative_prompt")],
    [sg.Button("Generar Imágenes")],
    [sg.Image(key="img1"), sg.Image(key="img2")],
    [sg.Text("Características de la Imagen 1:", key="text1")],
    [sg.Text("Características de la Imagen 2:", key="text2")],
    [sg.Button("Seleccionar Imagen 1"), sg.Button("Seleccionar Imagen
2"), sg.Exit()]
]
window = sg.Window("Generador de Imágenes con Stable Diffusion",
layout)
img_counter = 1
img1_selected = False
while True:
    event, values = window.read()
    if event == sg.WINDOW_CLOSED or event == "Exit":
        break
    if event == "Generar Imágenes":
        prompt = values["prompt"]
        negative_prompt = values["negative_prompt"]
        if len(prompt) == 0:
            sg.popup("Por favor, ingresa un prompt.")
            continue
        img1 = pipe(prompt, negative_prompt=negative_prompt,
height=512, width=512, num_inference_steps=5).images[0]
        img2 = pipe(prompt, negative_prompt=negative_prompt,
height=512, width=512, num_inference_steps=5).images[0]
        img1.save(f"img{img_counter}_1.png")
        img2.save(f"img{img_counter}_2.png")
        rgb_avg1, brillo_prom1, contraste_prom1 =
calcular_caracteristicas(img1)
```

```
        rgb_avg2, brillo_prom2, contraste_prom2 =
calcular_caracteristicas(img2)
        window["img1"].update(filename=f"img{img_counter}_1.png")
        window["img2"].update(filename=f"img{img_counter}_2.png")
        window["text1"].update(mostrar_caracteristicas(1, rgb_avg1,
brillo_prom1, contraste_prom1))
        window["text2"].update(mostrar_caracteristicas(2, rgb_avg2,
brillo_prom2, contraste_prom2))
    if event == "Seleccionar Imagen 1":
        sg.popup("Seleccionaste la Imagen 1")
        img1_selected = True
    if event == "Seleccionar Imagen 2":
        sg.popup("Seleccionaste la Imagen 2")
        img1_selected = False
        #Regenerar la primera imagen basada en la segunda imagen
        sg.popup("Regenerando Imagen 1 en base a las características de
Imagen 2...")
        img1 = pipe(prompt, negative_prompt=negative_prompt,
height=512, width=512, num_inference_steps=5).images[0]
        img1.save(f"img{img_counter}_1_regenerated.png")
        #Calcular nuevas características de la imagen regenerada
        rgb_avg1, brillo_prom1, contraste_prom1 =
calcular_caracteristicas(img1)
        #Actualizar la interfaz

window["img1"].update(filename=f"img{img_counter}_1_regenerated.png")
        window["text1"].update(mostrar_caracteristicas(1, rgb_avg1,
brillo_prom1, contraste_prom1))
    if img1_selected:
        sg.popup("Regenerando Imagen 2 en base a las características de
Imagen 1...")
        img2 = pipe(prompt, negative_prompt=negative_prompt,
height=512, width=512, num_inference_steps=5).images[0]
        img2.save(f"img{img_counter}_2_regenerated.png")
        rgb_avg2, brillo_prom2, contraste_prom2 =
calcular_caracteristicas(img2)

window["img2"].update(filename=f"img{img_counter}_2_regenerated.png")
        window["text2"].update(mostrar_caracteristicas(2, rgb_avg2,
brillo_prom2, contraste_prom2))
window.close()
for file in os.listdir():
    if file.endswith(".png"):
```

```
        os.remove(file)
```

Para que dé mejores resultados:
-Cambiar device a cuda.
-Cambiar “num_inference_steps=30” a más de 50 o 100.
-Aumentar resolución (1024x1024)
-Cambiar modelo.

Explicación powerpoint:
https://docs.google.com/presentation/d/1UK-wUNsYRyohfwh5iKDuIOimLDeo0yKgsqUJ8h7Pkno/edit#slide=id.g2d4c3faace4_1_81