Configurateur de switch:

Pendant mon stage, j'ai dû apprendre à utiliser les bibliothèques **Paramiko** et **Tkinter** afin de développer un configurateur de switch. Ce projet m'a demandé **environ deux semaines et quelques jours** d'apprentissage et de mise en place. Chaque jour, je découvrais de **nouvelles fonctionnalités** qui me permettaient d'améliorer mon code et de rendre mon programme plus efficace et intuitif.

Paramiko m'a été essentiel pour établir des connexions SSH sécurisées avec les équipements réseau.

```
def configure_switch(host, username, password, commands):
    try:
        client = paramiko.SSHClient()
        client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        client.connect(host, username=username, password=password, timeout=5)

        shell = client.invoke_shell()
        output = ""
        for command in commands.split('\n'):
            shell.send(command + "\n")
            shell.recv(1024)  # Attente de réponse du switch

        shell.send("exit\n")
        client.close()

        messagebox.showinfo("Succès", "Configuration appliquée avec succès !")
    except Exception as e:
        messagebox.showerror("Erreur", str(e))
```

```
def fetch_config(host, username, password):
    try:
        client = paramiko.SSHClient()
        client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        client.connect(host, username=username, password=password, timeout=5)

        shell = client.invoke_shell()
        shell.send("show running-config\n")
        output = shell.recv(5000).decode("utf-8")

        client.close()
        text_output.delete("1.0", tk.END)
        text_output.insert(tk.END, output)
    except Exception as e:
        messagebox.showerror("Erreur", str(e))
```

Grâce à cette bibliothèque, j'ai pu envoyer des commandes aux switches et récupérer des configurations de manière automatisée.
Cela m'a permis de mieux comprendre comment fonctionne la communication entre un script Python et un équipement réseau.

En parallèle, j’ai utilisé **Tkinter** pour créer une interface graphique conviviale, facilitant l’interaction avec mon configurateur. Au départ, cela me paraissait complexe, mais en expérimentant et en testant différentes options, j’ai réussi à concevoir une interface claire et ergonomique.

```
app = tk.Tk()
app.title("Switch Configurator")

# Labels et champs de saisie
tk.Label(app, text="Adresse IP du Switch:").pack()
entry_host = tk.Entry(app)
entry_host.pack()

tk.Label(app, text="Nom d'utilisateur:").pack()
entry_username = tk.Entry(app)
entry_username.pack()

tk.Label(app, text="Mot de passe:").pack()
entry_password = tk.Entry(app, show="*")
entry_password.pack()

tk.Label(app, text="Commandes de configuration:").pack()
text_commands = tk.Text(app, height=10, width=50)
text_commands.pack()

# Boutons
tk.Button(app, text="Appliquer Configuration", command=start_configuration).pack
tk.Button(app, text="Récupérer Configuration", command=retrieve_configuration).p

# Zone d'affichage de la configuration récupérée
tk.Label(app, text="Configuration Actuelle:").pack()
text_output = scrolledtext.ScrolledText(app, height=15, width=60)
text_output.pack()

app.mainloop()
```

Tout au long de cette période, j’ai beaucoup progressé en **programmation Python**, notamment en découvrant des concepts liés à l’automatisation et à la gestion des interfaces utilisateur. Ce projet m’a également appris à chercher des solutions par moi-même, à lire la documentation et à surmonter les difficultés techniques au fur et à mesure.

En fin de compte, cette expérience m’a apporté **de nouvelles compétences** et une meilleure compréhension des outils utilisés en administration réseau et en développement d’interfaces graphiques.

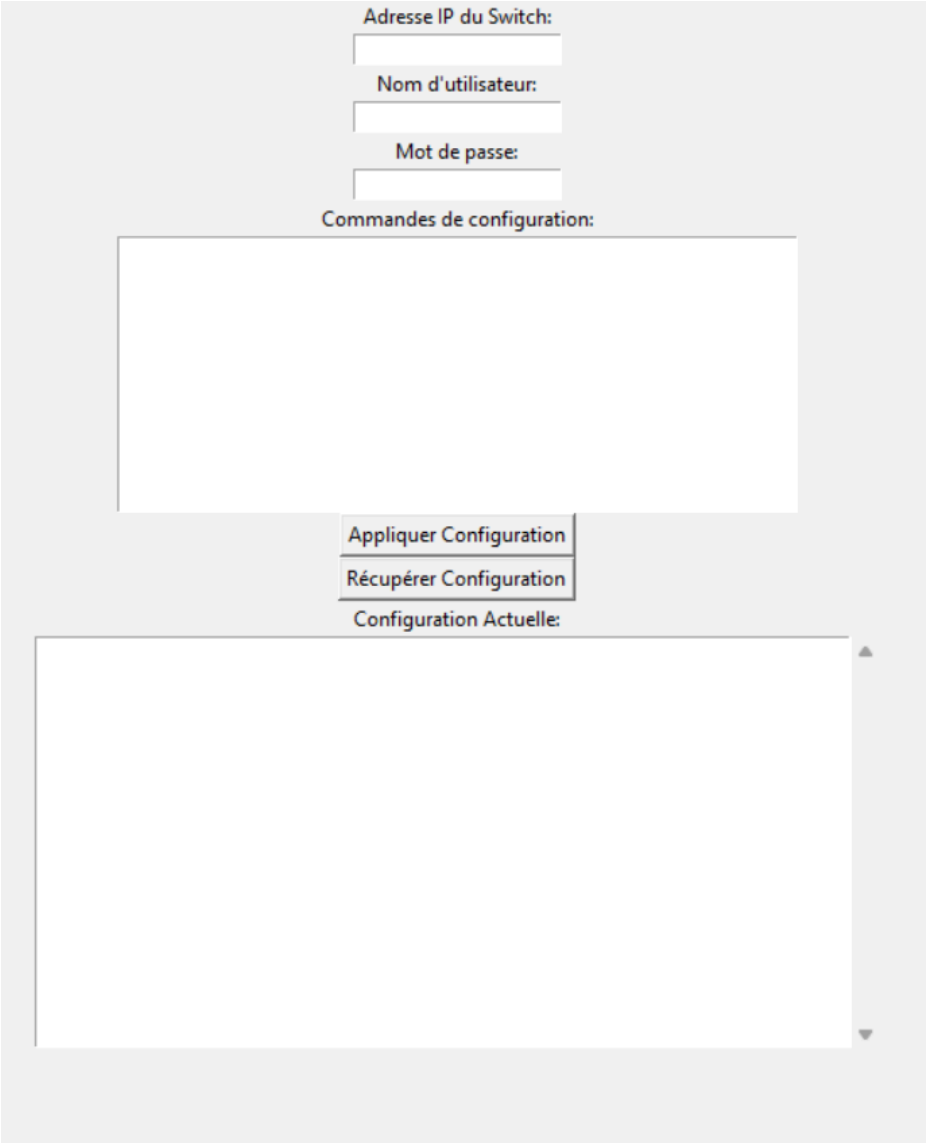

Schéma Réseau:

En plus de mon configurateur de switch, j'ai également développé un **simulateur de schéma réseau** en Python en utilisant **Tkinter** pour l'interface graphique. Ce logiciel permet de **créer, organiser et visualiser des réseaux** de manière interactive.

J'ai utilisé plusieurs bibliothèques pour enrichir les fonctionnalités :

- **Tkinter** pour la gestion des éléments graphiques et l'interaction utilisateur,
- **PIL (Pillow)** pour la manipulation des images,
- **ReportLab** pour l'exportation des schémas en **PDF**,
- **JSON** pour la sauvegarde et le chargement des configurations réseau,
- **ipaddress** pour la gestion et la validation des adresses IP.

```
import tkinter as tk
from tkinter import simpledialog, filedialog, messagebox, colorchooser
from PIL import Image, ImageTk, ImageGrab
from reportlab.pdfgen import canvas as pdf_canvas
from reportlab.lib.pagesizes import A4
import json
import os
import sys
import ipaddress
import math
from tkinter import ttk, PhotoImage
import subprocess
import platform
from datetime import datetime
```