



# Création d'un chat en réseau

Hadil  
Alyssa  
Abdoulaye





---

# INTERFACE EN LIGNE DE COMMANDE

## TECHNOLOGIES

- Python
  - Socket
  - Threading
- 

# TECHNOLOGIES

## PYTHON



## SOCKET

Socket permet la communication réseau entre des processus s'exécutant sur la même machine ou sur des machines distantes. Notre socket serveur utilise le protocole TCP.

## THREADING

Le module Threading permet de gérer des threads. Un thread est une séquence d'exécution indépendante qui peut être utilisée pour effectuer des tâches en arrière-plan tout en continuant à exécuter d'autres parties du programme.



# FONCTIONNEMENT

## CRÉATION DU SERVEUR – Server.py

- Configuration du serveur
- Fonctions gérant la réception et l'envoi des messages vers les différents terminaux

## CRÉATION DU CLIENT – Client.py

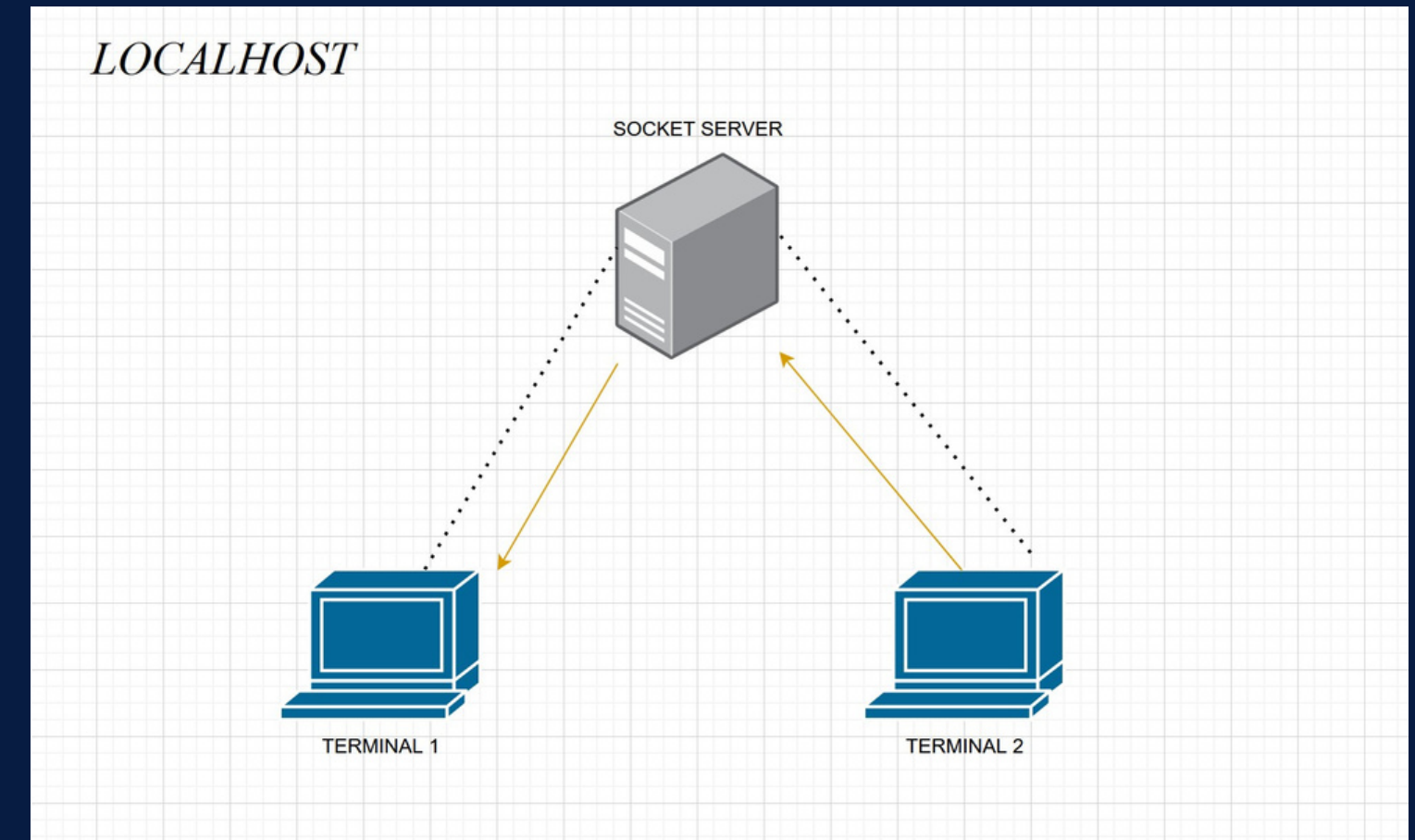
- Connexion au serveur
- Fonctions gérant l'envoi et la réception des messages grâce au serveur

```
PS C:\Users\USER-1\Documents\Fla
sk\ChatPython\Part_1> python ser
ver.py
server is running ...
connection with ('10.21.0.46', 6
2062)
b'Lola is connected '
server is running ...
connection with ('10.21.0.46', 6
2074)
b'Eric is connected '
server is running ...
█
```

```
PS C:\Users\USER-1\Documents\Fla
sk\ChatPython\Part_1> python c
lient.py
choose a nickname Lola
you are connected
Eric : Bonjour
Salut Eric !
Lola : Salut Eric !
█
```

```
PS C:\Users\USER-1\Documents\Fla
sk\ChatPython\Part_1> python c
lient.py
choose a nickname Eric
you are connected
Bonjour
Eric : Bonjour
Lola : Salut Eric !
█
```

# FONCTIONNEMENT



## SERVEUR

- Création d'un serveur utilisant le protocole TCP
- Configuration avec l'adresse IP et le port
- Fonction broadcast, handle\_client, receive :
- Broadcast : récupère le message en paramètre et l'envoie dans le terminaux des clients connectés
- Handle Client : Récupère le message envoyé par le client en paramètre et appelle la fonction broadcast
- Receive : Fonction principale qui gère la connexion des clients

## CLIENT


- Initie la connexion au serveur grâce à l'adresse IP
- Fonction client\_receive, client\_send
- Client\_Receive : récupère le message envoyé par le serveur et agit en fonction du message reçu
- Client\_Send : Envoie le message souhaité au serveur

---



# Chat multi-utilisateur avec interface graphique

Cette partie consiste à la création d'une applications permettant à d'autres utilisateurs de communiquer entre eux dans différentes pages.





# TECHNOLOGIES

## PYTHON



## FLASK

Flask est un framework web minimaliste et flexible pour Python, conçu pour être simple à utiliser et facile à apprendre. Il est extensible grâce à son système d'extensions, ce qui permet d'ajouter facilement des fonctionnalités supplémentaires.

## SOCKETIO

Socket.IO est une bibliothèque JavaScript qui permet la communication bidirectionnelle en temps réel entre les clients web et les serveurs.

# CRÉATION DU SERVEUR EN FLASK

```
from flask import Flask, render_template
from flask_socketio import SocketIO
HOST = "192.168.235.135"
PORT = 5000
app = Flask(__name__, template_folder='templates')
socketio = SocketIO(app)

@app.route('/')
def index():
    return render_template('index.html')

@socketio.on('message')
def handle_message(message):
    print('Message received:', message)
    socketio.emit('message', message)

if __name__ == '__main__':
    socketio.run(app, HOST)
```