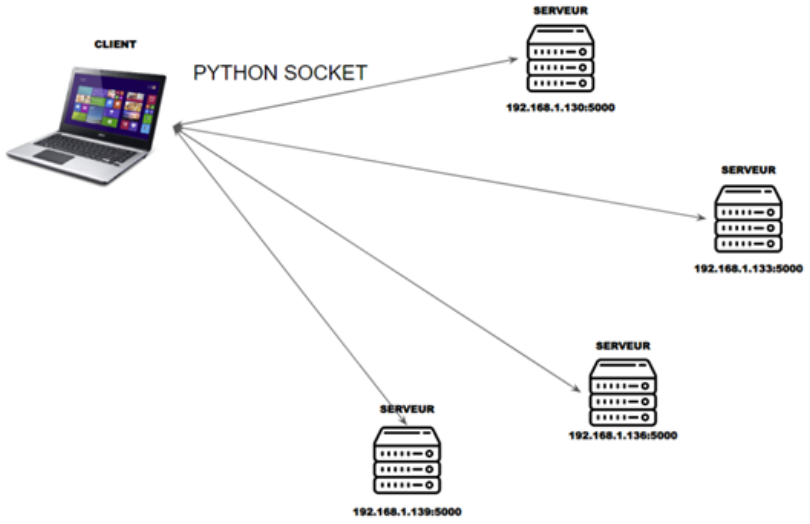
SAE 3.02
Technical Document

LAMINE ELIAS

To begin with, the purpose of our SAE is to be able to communicate between servers and a client. For this we will create 2 files based on the use of the "socket" and "thread" for communication between the two. All this will be seen directly for the client on a graphical interface that the client will have the right to access.

The SAE project will take place over a period of 4 weeks with dedicated hours of approximately 45 hours. This is a one-off job to do with the possibility that designer teachers can help fix some issues or bugs in the code that each student will create.

The chosen topology:



Server launch:

```python
import socket
import os
import sys, subprocess
import psutil

host = "127.0.0.1"
# port = int(input("port:"))
port = 8000
```

We have below the code which allows us to put the parameters for the connection of the server with the port and the ip defined as for example here the ip=127.0.0.1 and the port =8000.

```python
while True:                                                    # ⚠2 ⚠33 ✓61
    server = socket.socket()  # Création d'un canal de communication perme
    server.bind((host, port))
    server.listen(1)
    while True:
        print("En attente d'un client")
        conn, address = server.accept()  # ACCEPT
        print(f"Un client s'est connecté à partir de {address}")
        data = ""
```

Here we have the code that will allow us to create the communication channel.it will allow the launch of our server.
I created a "while True" loop to create the communication channel and will associate the host and the port then it will enter the second "while true" loop and the server will be waiting for the client.

Server and client status commands

```
elif data.lower() == "disconnect":
    print("deconnection du client au server")
    conn.send("disconnect".encode())
    break

#fonction kill
elif data.lower() == "kill":
    conn.send("disconnect".encode())
    conn.close()
    server.close()
    sys.exit(0)


elif data.lower() == "reset":
    # server = socket.socket()
    # server.bind((host,port))
    # server.listen(1)
    # print(f"{host}-{port}")
    conn.send("reset".encode())
    conn.close()
    server.close()
```

This part of my code represents the status commands of the server or the client or both.The "kill" command for example stops the server but lets the client run. We also have the "disconnect" which disconnects the client so the server will look for a client then and we have the reset command which will restart the client and the server.

So I used conditions so that if our client sends a message that is not equal to kill for example then it will not enter this kill condition and will not do what this condition plans to do which is to close d 'shut down the server

Miscellaneous orders

```python
def Hostname():
    cmd = socket.gethostname()
    return cmd


def ipconfig():
    host = socket.gethostname()
    cmd = socket.gethostbyname(host)
    return cmd


def ram():
    cmd = str(
        f"-La memoire total:{psutil.virtual_memory()[0]} octet\n-la memoi
    return cmd


def Os():
    cmd = sys.platform
    if cmd == "win32":
        cmd = subprocess.check_output("ver", shell=True).decode()
        return cmd
```

Here we can see the creation of different python functions such as OS or Ram,ip,name.
We will call these functions later if the commands sent by the client match if it does not match then we will have a return message that will tell us "invalid command".

```python
def server_socket():
    while True:
        server = socket.socket()  # Création d'un canal de communication perm
        server.bind((host, port))
        server.listen(1)
        while True:
            print("En attente d'un client")
            conn, address = server.accept()  # ACCEPT
            print(f"Un client s'est connecté à partir de {address}")
            data = ""
            while data != "bye" or data != "arret":
                try:
                    data = conn.recv(1024).decode()  # RECEIV
                    if data is not None:
                        if data.lower() == 'ip':
                            res = ipconfig()
                            conn.send(res.encode())

                        elif data.lower() == 'name':
                            res = Hostname()
                            conn.send(res.encode())

                        elif data.lower()[0:4] == "ping":
                            res = ping(data)
                            conn.send(res.encode())

                        elif data[0:4].lower() == "dos":#7 envrion
                            if sys.platform == 'win32':#
```

Customer part

We had to create an interface in the client file
In the client file we can see for example here in this piece of code a part of the graphic interface that we want to make appear like the "ok" button.

```python
class baseDeDonnee(QWidget):
    def __init__(self, parent):
        super().__init__()
        self.__parent = parent
        self.grid = QGridLayout()
        self.setLayout(self.grid)
        self.__table_csv = QTableWidget()
        self.__ip_edit = QLineEdit()
        self.__port_edit = QLineEdit()
        self.__confirm = QPushButton('OK')
        self.__confirm.clicked.connect(self.__ajouter_ligne)
        self.grid.addWidget(self.__table_csv, 0, 0, 1, 3)
        self.grid.addWidget(self.__ip_edit, 1, 0)
        self.grid.addWidget(self.__port_edit, 1, 1)
        self.grid.addWidget(self.__confirm, 1, 2)
        self.__table_csv.setRowCount(0)
        self.__table_csv.setColumnCount(1)
```

We have created different methods with conditions.

```python
def __lancement(self):
    client_socket = socket()
    HOST = self.__ip.text()
    PORT = self.__port.text()
    if PORT.isdigit():
        PORT = int(PORT)
        try:
            client_socket.connect((HOST, PORT))
            client_socket.setblocking(False)
            tab = connection(client_socket, self)
            self.__tabwidget.addTab(tab, str(HOST) + ':' + str(PORT))
        except:
            self.show_error(f'Erreur lors de la connection vers {HOST}:{PORT}')
    else:
        self.show_error('Veuillez insérer un numéro pour le port!')
```

Here we have a method called "__lancement" it will manage the creation of the connection to the server using the parameters inserted into the form by the user. Here for example if our port is not an int we will have a error message to warn us that our port is not an int and that we must therefore insert the correct port in int.