SAE 3.02 – Development of project

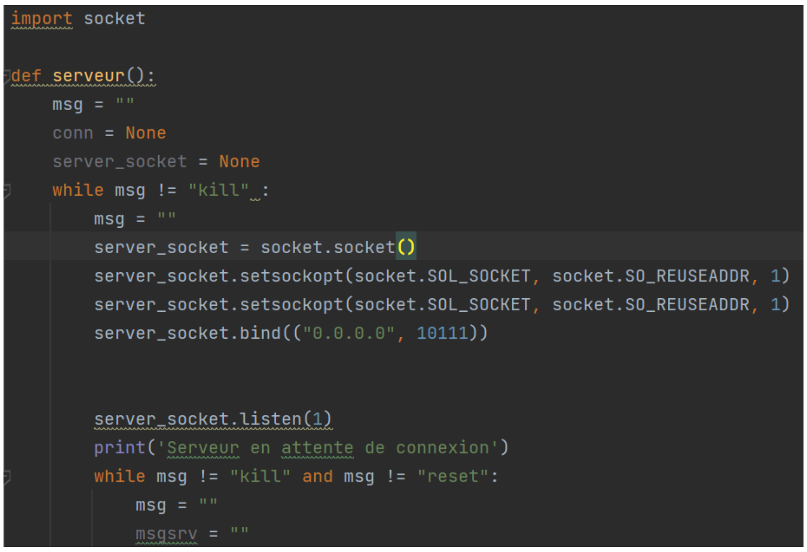
Objective of Project:

Achievement of client-server system allowing to communicate at distance and to send of system commands. The implementation of a graphic interface allowing to interrogate the server on his status and to send of commands allowing to have of information on the distant server.

My achievements:

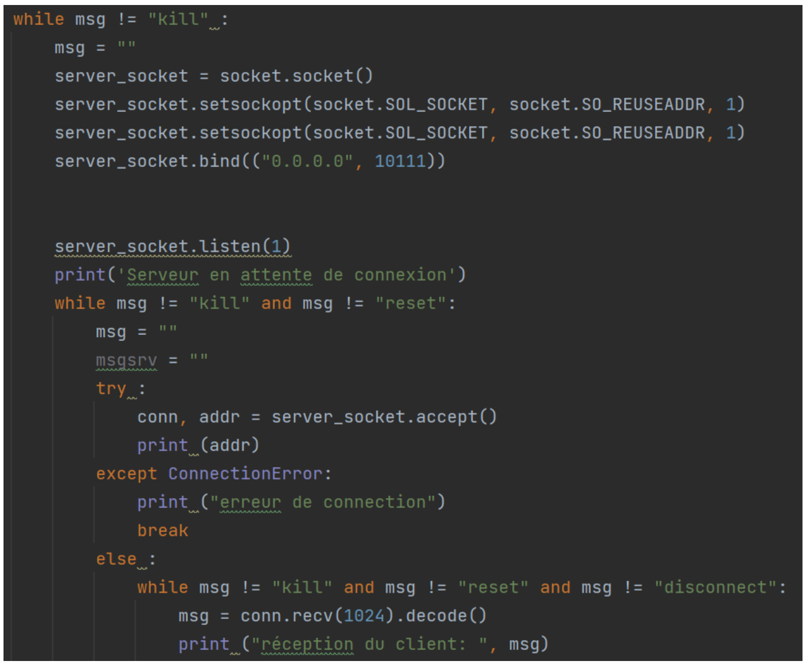
I have implementation 1 file for the server, 1 for the client and 1 graphic interface allowing to display of information on the server in local thanks to the request of the client.

1. Implementation server with the socket



```
import socket

def serveur():
    msg = ""
    conn = None
    server_socket = None
    while msg != "kill" :
        msg = ""
        server_socket = socket.socket()
        server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        server_socket.bind(("0.0.0.0", 10111))


        server_socket.listen(1)
        print('Serveur en attente de connexion')
        while msg != "kill" and msg != "reset":
            msg = ""
            msgsrv = ""
```

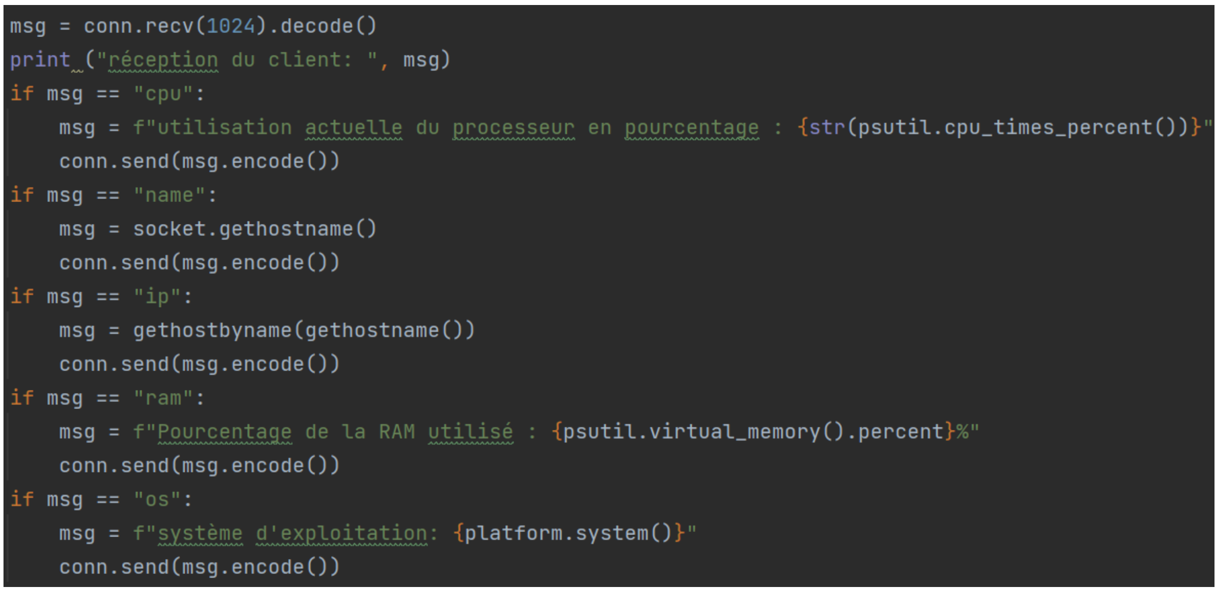
- To connect the server, we need of an IP address and port of connection to connect with the server.



```
while msg != "kill" :
    msg = ""
    server_socket = socket.socket()
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_socket.bind(("0.0.0.0", 10111))


    server_socket.listen(1)
    print('Serveur en attente de connexion')
    while msg != "kill" and msg != "reset":
        msg = ""
        msgsrv = ""
        try :
            conn, addr = server_socket.accept()
            print (addr)
        except ConnectionError:
            print ("erreur de connection")
            break
        else :
            while msg != "kill" and msg != "reset" and msg != "disconnect":
                msg = conn.recv(1024).decode()
                print ("réception du client: ", msg)
```

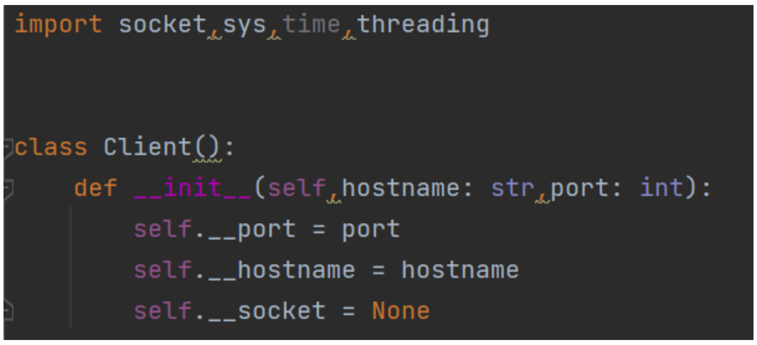
- By treating the return kill, disconnect, reset by setting conditions and an exception for the connection error.

- I have indicated at my server what information it should send in receive the message of client (cpu, name, ip, ram, os).



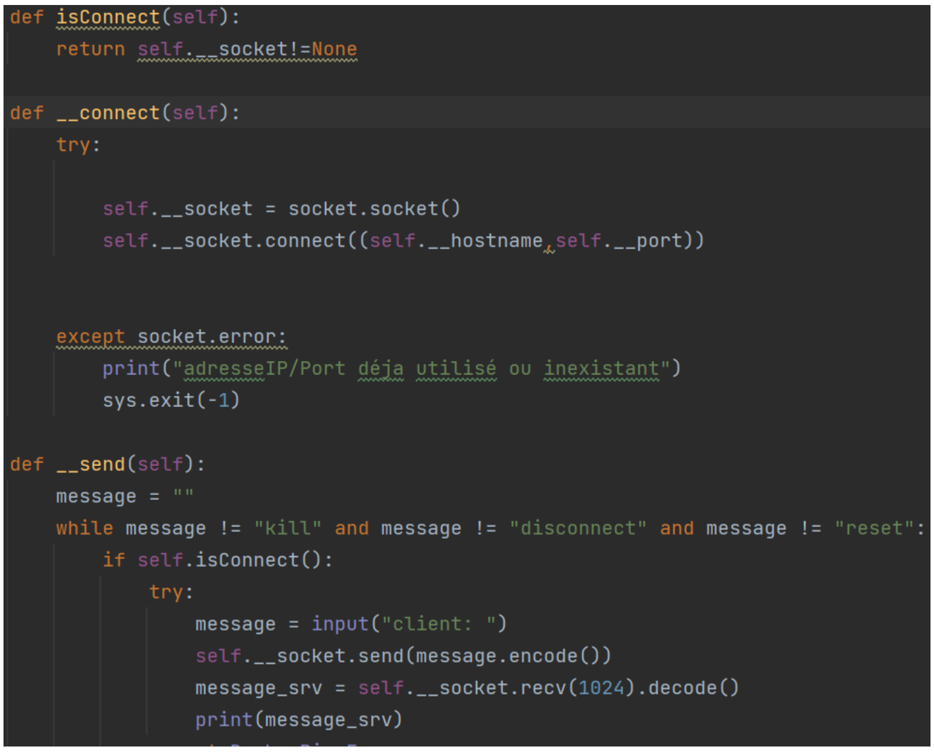
```
msg = conn.recv(1024).decode()
print ("réception du client: ", msg)
if msg == "cpu":
    msg = f"utilisation actuelle du processeur en pourcentage : {str(psutil.cpu_times_percent())}"
    conn.send(msg.encode())
if msg == "name":
    msg = socket.gethostname()
    conn.send(msg.encode())
if msg == "ip":
    msg = gethostbyname(gethostname())
    conn.send(msg.encode())
if msg == "ram":
    msg = f"Pourcentage de la RAM utilisé : {psutil.virtual_memory().percent}%"
    conn.send(msg.encode())
if msg == "os":
    msg = f"système d'exploitation: {platform.system()}"
    conn.send(msg.encode())
```

2. <u>Implementation client with the thread</u>
    - Add of an address at client and a port of connection to connect with the server.



```
import socket,sys,time,threading


class Client():
    def __init__(self,hostname: str,port: int):
        self.__port = port
        self.__hostname = hostname
        self.__socket = None
```

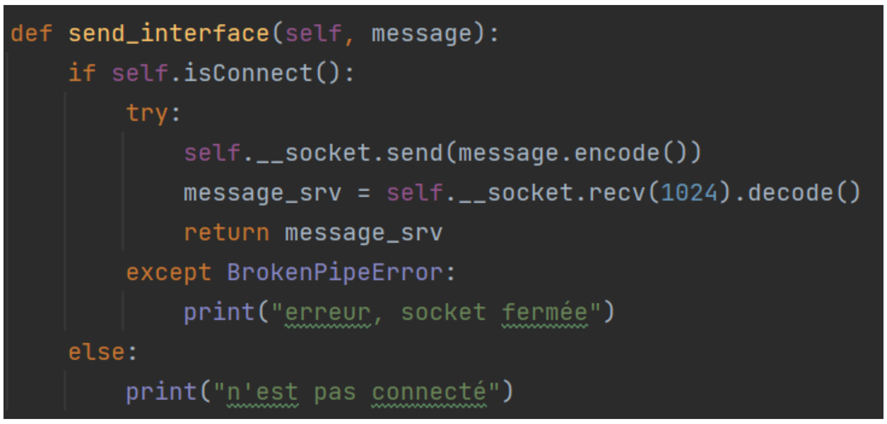
- I have separated my code in multiple functions



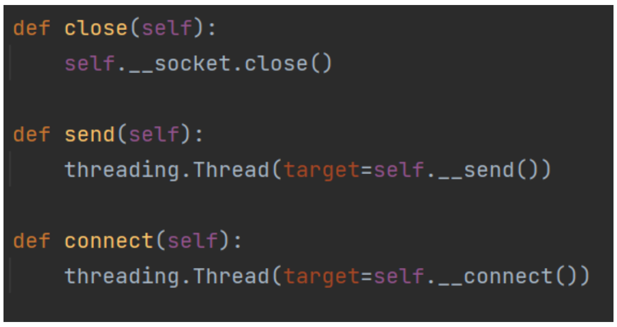
```
def isConnect(self):
    return self.__socket!=None

def __connect(self):
    try:

        self.__socket = socket.socket()
        self.__socket.connect((self.__hostname,self.__port))


    except socket.error:
        print("adresseIP/Port déja utilisé ou inexistant")
        sys.exit(-1)

def __send(self):
    message = ""
    while message != "kill" and message != "disconnect" and message != "reset":
        if self.isConnect():
            try:
                message = input("client: ")
                self.__socket.send(message.encode())
                message_srv = self.__socket.recv(1024).decode()
                print(message_srv)
```

- “isConnection” to verify the connection of client
- “__connect” to connect the client with his address and port of connection.
- “send” for the send and receive in treating the exception

- A function other to send and receive but for the interface and an exception



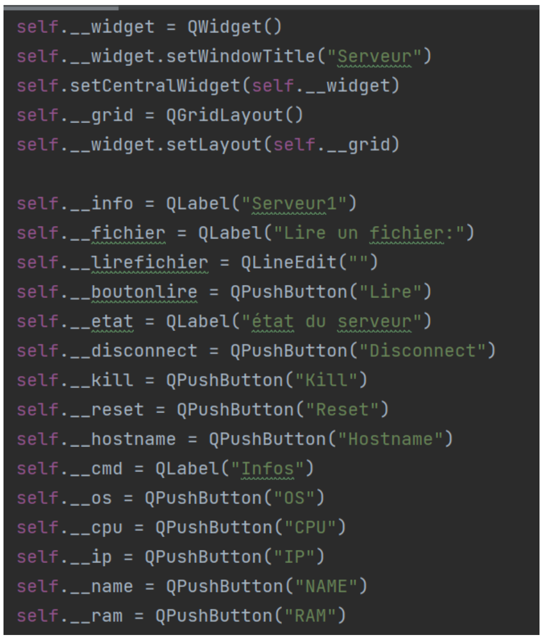
```python
def send_interface(self, message):
    if self.isConnect():
        try:
            self.__socket.send(message.encode())
            message_srv = self.__socket.recv(1024).decode()
            return message_srv
        except BrokenPipeError:
            print("erreur, socket fermée")
    else:
        print("n'est pas connecté")
```



```python
def close(self):
    self.__socket.close()

def send(self):
    threading.Thread(target=self.__send())

def connect(self):
    threading.Thread(target=self.__connect())
```

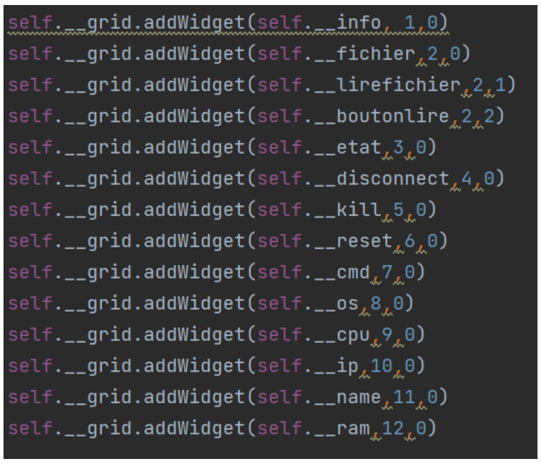
- Call of the functions overhead to use in the file of the interface.

3. <u>Main interface</u>



```python
self.__widget = QWidget()
self.__widget.setWindowTitle("Serveur")
self.setCentralWidget(self.__widget)
self.__grid = QGridLayout()
self.__widget.setLayout(self.__grid)

self.__info = QLabel("Serveur1")
self.__fichier = QLabel("Lire un fichier:")
self.__lirefichier = QLineEdit("")
self.__boutonlire = QPushButton("Lire")
self.__etat = QLabel("état du serveur")
self.__disconnect = QPushButton("Disconnect")
self.__kill = QPushButton("Kill")
self.__reset = QPushButton("Reset")
self.__hostname = QPushButton("Hostname")
self.__cmd = QLabel("Infos")
self.__os = QPushButton("OS")
self.__cpu = QPushButton("CPU")
self.__ip = QPushButton("IP")
self.__name = QPushButton("NAME")
self.__ram = QPushButton("RAM")
```

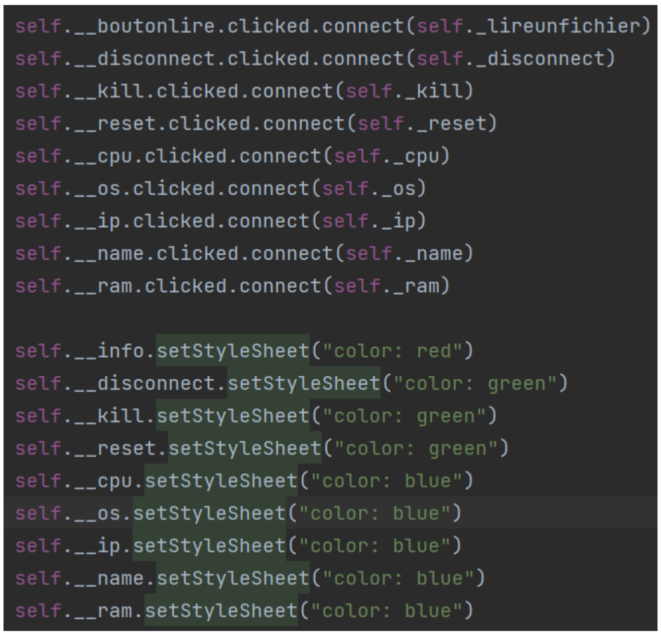
- Initialization of title of the window and of the elements to display in the interface



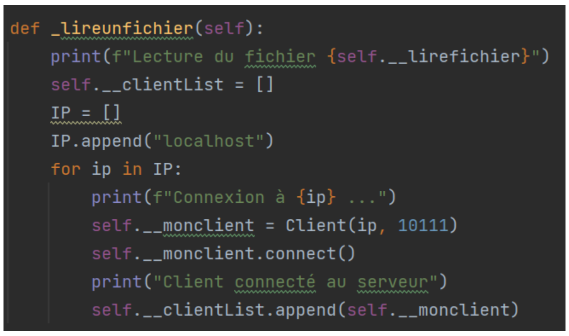
```python
self.__grid.addWidget(self.__info, 1,0)
self.__grid.addWidget(self.__fichier,2,0)
self.__grid.addWidget(self.__lirefichier,2,1)
self.__grid.addWidget(self.__boutonlire,2,2)
self.__grid.addWidget(self.__etat,3,0)
self.__grid.addWidget(self.__disconnect,4,0)
self.__grid.addWidget(self.__kill,5,0)
self.__grid.addWidget(self.__reset,6,0)
self.__grid.addWidget(self.__cmd,7,0)
self.__grid.addWidget(self.__os,8,0)
self.__grid.addWidget(self.__cpu,9,0)
self.__grid.addWidget(self.__ip,10,0)
self.__grid.addWidget(self.__name,11,0)
self.__grid.addWidget(self.__ram,12,0)
```

- Display of the elements in the window

- Connection of the buttons at their corresponding function and append style color

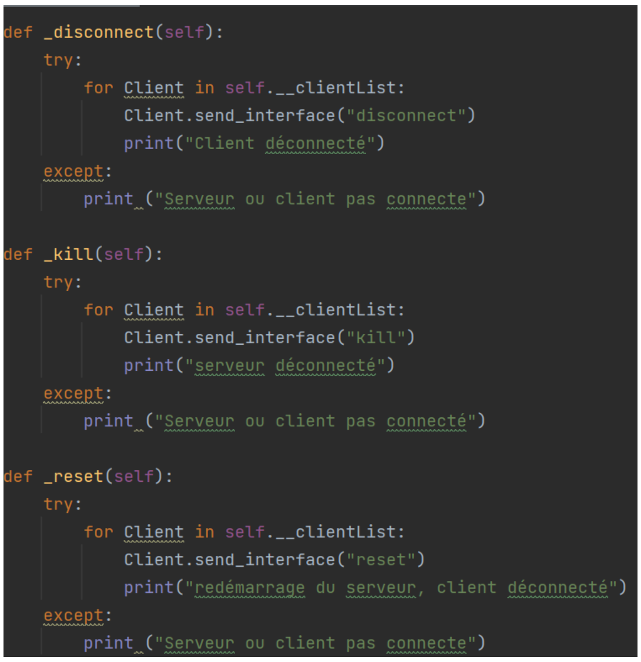


```
self.__boutonlire.clicked.connect(self._lireunfichier)
self.__disconnect.clicked.connect(self._disconnect)
self.__kill.clicked.connect(self._kill)
self.__reset.clicked.connect(self._reset)
self.__cpu.clicked.connect(self._cpu)
self.__os.clicked.connect(self._os)
self.__ip.clicked.connect(self._ip)
self.__name.clicked.connect(self._name)
self.__ram.clicked.connect(self._ram)

self.__info.setStyleSheet("color: red")
self.__disconnect.setStyleSheet("color: green")
self.__kill.setStyleSheet("color: green")
self.__reset.setStyleSheet("color: green")
self.__cpu.setStyleSheet("color: blue")
self.__os.setStyleSheet("color: blue")
self.__ip.setStyleSheet("color: blue")
self.__name.setStyleSheet("color: blue")
self.__ram.setStyleSheet("color: blue")
```



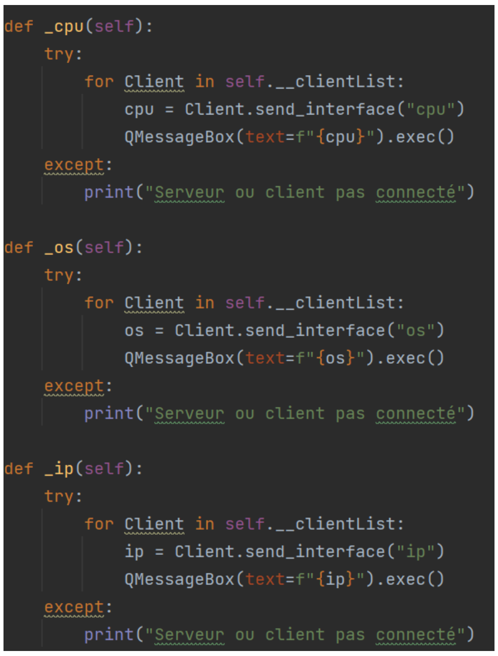
```
def _lireunfichier(self):
    print(f"Lecture du fichier {self.__lirefichier}")
    self.__clientList = []
    IP = []
    IP.append("localhost")
    for ip in IP:
        print(f"Connexion à {ip} ...")
        self.__monclient = Client(ip, 10111)
        self.__monclient.connect()
        print("Client connecté au serveur")
        self.__clientList.append(self.__monclient)
```

- To connect the client at my interface



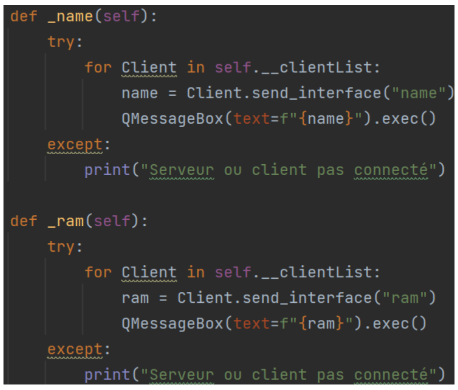
```
def _disconnect(self):
    try:
        for Client in self.__clientList:
            Client.send_interface("disconnect")
            print("Client déconnecté")
    except:
        print ("Serveur ou client pas connecte")

def _kill(self):
    try:
        for Client in self.__clientList:
            Client.send_interface("kill")
            print("serveur déconnecté")
    except:
        print ("Serveur ou client pas connecté")

def _reset(self):
    try:
        for Client in self.__clientList:
            Client.send_interface("reset")
            print("redémarrage du serveur, client déconnecté")
    except:
        print ("Serveur ou client pas connecte")
```

- Functions of the button click event. Sending of a message of client at server and reception of message of server in interface.



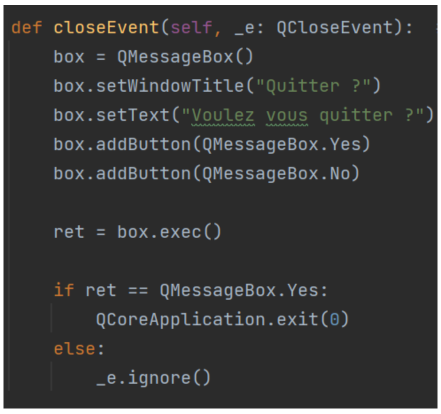
```python
def _cpu(self):
    try:
        for Client in self.__clientList:
            cpu = Client.send_interface("cpu")
            QMessageBox(text=f"{cpu}").exec()
    except:
        print("Serveur ou client pas connecté")

def _os(self):
    try:
        for Client in self.__clientList:
            os = Client.send_interface("os")
            QMessageBox(text=f"{os}").exec()
    except:
        print("Serveur ou client pas connecté")

def _ip(self):
    try:
        for Client in self.__clientList:
            ip = Client.send_interface("ip")
            QMessageBox(text=f"{ip}").exec()
    except:
        print("Serveur ou client pas connecté")
```

- Implementation of exception to verify the connection



```python
def _name(self):
    try:
        for Client in self.__clientList:
            name = Client.send_interface("name")
            QMessageBox(text=f"{name}").exec()
    except:
        print("Serveur ou client pas connecté")

def _ram(self):
    try:
        for Client in self.__clientList:
            ram = Client.send_interface("ram")
            QMessageBox(text=f"{ram}").exec()
    except:
        print("Serveur ou client pas connecté")
```

- Append an event of close of the window with the cross.



```python
def closeEvent(self, _e: QCloseEvent):
    box = QMessageBox()
    box.setWindowTitle("Quitter ?")
    box.setText("Voulez vous quitter ?")
    box.addButton(QMessageBox.Yes)
    box.addButton(QMessageBox.No)

    ret = box.exec()

    if ret == QMessageBox.Yes:
        QCoreApplication.exit(0)
    else:
        _e.ignore()
```