Syrian Arab Republic
Lattakia - Tishreen University
Department of Communication and electrical engineering
5th , Network Programming : Homework No1

الجمهورية العربية السورية
اللاذقية -جامعة تشرين
كلية الهندسة الكهربائية والميكانيكية
قسم هندسة الاتصالات والالكترونيات
السنة الخامسة: وظيفة1 برمجة شبكات

Name:Yara Shahoud, Number:2122, Submitted To GitHub:

https://github.com/YaraShahoud/Network_Programming_Homework_No2_2024.git

Question 1: Bank ATM Application with TCP Server/Client and Multi-threading

Project Description:

Build a TCP server and client Bank ATM application using Python. The server should handle

multiple client connections simultaneously using multi-threading. The application should

allow clients to connect, perform banking operations (such as check balance, deposit, and

withdraw), and receive their updated account status upon completion.

Requirements:

A. The server should be able to handle multiple client connections concurrently.

B. The server should maintain a set of pre-defined bank accounts with balances.

C. Each client should connect to the server and authenticate with their account details.

D. Clients should be able to perform banking operations: check balance, deposit money, and withdraw money.

E. The server should keep track of the account balances for each client.

F. At the end of the session, the server should send the final account balance to each client

قمت بالبحث في المواقع المرفقة ومرجع المادة وتم التوصل الى عدة نقاط أساسية مفيدة في تطوير البرنامج:

-نحتاج لبرمجة المقابس باستخدام socket حيث واجهت صعوبة في المزامنة بشكل صحيح بالرسائل بين المخدم والزبون.

-نحتاج لبرمجة الى التفرع باستخدام multithreading حيث عانيت من الحاجة الى ضمان تعديل البيانات بشكل صحيح وعدم جمود النظام.

-نحتاج الى قاعدة بيانات لتخزين المعلومات حيث تم استخدام ملفات نصيية.

كود المخدم الرئيسي:

```
import threading
import socket
from ProjetPSR.CompteManager import findRefCompte_login, readCompte

#SERVER = socket.gethostbyname(socket.gethostname())
from ProjetPSR.FactureManager import readFacture
from ProjetPSR.TransactionManager import EffectuerTransaction

SERVER = "192.168.1.14"
PORT = 9090
ADDR = (SERVER, PORT)
FORMAT = 'utf-8'
HEADER = 64

DISCONNECT_MESSAGE = "!DISCONNECT"
ANSWER_MESSAGE = "!ANSWER"

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(ADDR)

#Liste des connections
```

```
connections = []
#Liste des comptes en train d'effectuer des transactions
comptes_actives = []


#Connection----------------------------------------
#Envoyer message
def send_message(conn, msg):
    message = msg.encode(FORMAT)
    msg_length = len(message)
    send_length = str(msg_length).encode(FORMAT)
    send_length += b' ' * (HEADER - len(send_length))
    conn.send(send_length)
    conn.send(message)
#Recevoir message
def recieve_message(conn):
    send_message(conn, ANSWER_MESSAGE)
    msg_length = conn.recv(HEADER).decode(FORMAT)
    if msg_length:
        msg_length = int(msg_length)
        msg = conn.recv(msg_length).decode(FORMAT)
    return msg


#Semaphore-----------------------------------------
def lock(conn, compte):
    if compte.refCompte in comptes_actives:
        msg = "Ce compte a une transaction en cours! Veuillez
patientez!"
        send_message(conn, msg)
    while True:
        if compte.refCompte not in comptes_actives:
            break
    comptes_actives.append(compte.refCompte)
def unlock(compte):
    comptes_actives.remove(compte.refCompte)


def login(conn):
    msg = "Bienvenue chez MEZGAR & REDISSI BANKING ! \n Cher client !
Veuillez saisir votre numéro de compte :"
    send_message(conn, msg)
    refCompte = recieve_message(conn)
    refCompte.strip()
    if refCompte.lower() == "00000":  # admin
        token = {"status": True, 'data': "admin"}
    else:
        token = findRefCompte_login(refCompte)
    print(f"{refCompte} ")
```

```
    return token


def menu(conn):
    choix = '0'
    while choix not in '12345':
        msg = "Veuillez taper votre choix ( 1 - 5 ) "
        send_message(conn, msg)
        msg = "1 ----- Consulter compte ----- "
        send_message(conn, msg)
        msg = "2 ----- Debiter ----- "
        send_message(conn, msg)
        msg = "3 ----- Crediter ----- "
        send_message(conn, msg)
        msg = "4 ----- Recevoir facture ----- "
        send_message(conn, msg)
        msg = "5 XXXXX Deconnexion XXXXX"
        send_message(conn, msg)
        choix = recieve_message(conn)
        print(choix)
        if choix not in '12345':
            msg = "Choix invalide !"
            send_message(conn, msg)
    return choix

def handle_menu(conn, compte, choix):
    match choix:
        case '1':
            compte = readCompte(compte.refCompte)
            send_message(conn, str(compte))
        case '2':
            lock(conn, compte)
            msg = "Débit-------------------"
            send_message(conn, msg)
            msg = "Montant : "
            send_message(conn, msg)
            montant = int(recieve_message(conn))
            print('MOOOOOOOONTANT')
            print(montant)
            while montant < 0:
                msg = "Vous devez introduire un montant positif"
                send_message(conn, msg)
                montant = int(recieve_message(conn))

            operation = EffectuerTransaction(compte.refCompte,
"retrait", montant)
            if operation == True:
                msg = f"Retrait de {montant}DT effectue avec success!"
                send_message(conn, msg)
```

```
                facture = readFacture(compte.refCompte)
                msg = "Votre Facture"
                send_message(conn, msg)
                send_message(conn, str(facture))
            else:
                msg = "ERREUR: Retrait Echoue! Vous avez depassez le
plafond !"
                send_message(conn, msg)
            unlock(compte)
        case '3':
            lock(conn, compte)
            msg = "Donnez le montant a ajouter ?"
            send_message(conn, msg)
            montant = int(recieve_message(conn))
            while montant < 0:
                msg = "Vous devez introduire un montant positif"
                send_message(conn, msg)
                montant = int(recieve_message(conn))
            operation = EffectuerTransaction(compte.refCompte, "depot",
montant)
            if operation == True:
                msg = f"Ajout de {montant}DT effectue avec success!"
            elif operation == False:
                msg = "ERREUR: Ajout Echoue! Reessayez !"
            send_message(conn, msg)
            unlock(compte)
        case '4':
            facture = readFacture(compte.refCompte)
            send_message(conn, str(facture))


def handle_client(conn, addr):
    print(f"[NEW CONNECTION] {addr} connected.")
    connected = True
    token = login(conn)
    if token['status']:
        compte = token['data']
        try:
            send_message(conn, f"Bienvenue {token['data']}")
        except:
            send_message(conn, f"Bienvenue {token['data']}")
    else:
        send_message(
            conn, "Informations Invalides! Vous avez etes deconnecte!")
        send_message(conn, DISCONNECT_MESSAGE)
        connected = False
        connections.remove(conn)
        print(f"[{addr}] was disconnected due to invalid info!")
    while connected:
```

```
            #if compte == 'admin':
             #   option = send_admin_menu(conn)
             #   if option == '5':
              #      send_message(conn, DISCONNECT_MESSAGE)
               #     connected = False
                #    connections.remove(conn)
                 #   print(f"[{addr}] has disconnected!")
                  #  break
                #handle_admin_option(conn, option)
            #else:
            if 'admin' != compte:
                choix = menu(conn)
                print("hedhi")
                print(choix)
                if choix == '5':
                    send_message(conn, DISCONNECT_MESSAGE)
                    connected = False
                    connections.remove(conn)
                    print(f"[{addr}] has disconnected!")
                    break
                handle_menu(conn, compte, choix)
        conn.close()

    def demarrer():
        server.listen()
        print(f"Bienvenu chez MEZGAR and REDISSI BANKING ")
        print(f"[LISTENING] Server is listening on {SERVER}")
        while True:
            conn, addr = server.accept()
            connections.append(conn)
            thread = threading.Thread(target=handle_client, args=(conn,
    addr))
            thread.start()
            print(f"[ACTIVE CONNECTIONS] {threading.active_count() - 1}")


    print("[STARTING] server is starting...")
    demarrer()
```

كود الزبون:

```
import socket
import threading
```

```
SERVER = "192.168.1.14"
PORT = 9090
ADDR = (SERVER, PORT)
FORMAT = 'utf-8'
HEADER = 64
DISCONNECT_MESSAGE = "!DISCONNECT"
ANSWER_MESSAGE = "!ANSWER"

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(ADDR)


def send_msg(msg):
    message = msg.encode(FORMAT)
    msg_length = len(message)
    send_length = str(msg_length).encode(FORMAT)
    send_length += b' ' * (HEADER - len(send_length))
    client.send(send_length)
    client.send(message)


def recieve_msg():
    while True:
        msg_length = client.recv(HEADER).decode(FORMAT)
        if msg_length:
            msg_length = int(msg_length)
            msg = client.recv(msg_length).decode(FORMAT)
            if msg == ANSWER_MESSAGE:
                msg = input()
                send_msg(msg)
            elif msg == DISCONNECT_MESSAGE:
                break
            else:
                print(msg)


thread_talk = threading.Thread(target=recieve_msg, args=())
thread_talk.start()
```

Question 2: Simple Website Project with Python Flask Framework (you have choice to use Django or any Other Deferent Useful Python Project “from provide Project Links”)

Create a simple website with multiple pages using Flask, HTML, CSS, and Bootstrap. The website should demonstrate your understanding of web design principles.

تم استخدام منصة Django في تطوير الموقع حيث قمت بتطوير عدة تطبيقا لتعلم البيئة وتم استخدام عدة تقنيات وواجهات في ذلك.