UNIVERSITETI I PRISHTINËS

FAKULTETI I INXHINIERISË ELEKTRIKE DHE KOMPJUTERIKE

DEPARTAMENTI INXHINIERI KOMPJUTERIKE

PROJEKTI: DIZAJNIMI KLIENT-SERVER

LËNDA: RRJETA KOMPJUTERIKE

Punoi: Alberiana Tofaj

NrID: 190718100030

Mentor: Prof. Dr. Blerim Rexha

Msc. Haxhi Lajqi

Maj 2021

Abstrakt

Ky raport përmban dizajnimin e protokollit FIEK për komunikimin klient-server sipas TCP protokollit dhe UDP protokollit në kuadër të lëndës Rrjeta Kompjuterike. Idea kryesore e këtij raporti është dizajnimi i një programi komunikues klient-server në gjuhën programuese python ku klienti mund të bëjë kërkesa dhe varësisht nga kërkesa e klientit serveri kthen përgjigjja. Zgjidhja e metodave bëhet në server ndërsa thirrja e tyre për tu shfaqur bëhet tek klienti. Po ashtu është implementuar edhe funksioni i cili lejon komunikimin e disa klientëve me një server në të njëjtën kohë.

Përmbajtja

Hyrje

Qëllimi kryesor i këtij projekti është kuptimi protokolleve që lidhen me arkitekturën klient-server. Objektivat e këtij projekti janë që të bëhet dizajnimi, implementimi dhe testimi i programit klient dhe server që implementohet në versionet TCP dhe UDP dhe po ashtu përdorimi i thread-ave. implementimi në versionet TCP dhe UDP mundësohet përmes soketeve. Programimi i soketave është lidhje në mes të dy soketave apo rrjetave qe te komunikojnë me njëra-tjetrën. Njeri soket është duke pritur ne një port te caktuar deri sa tjetri soket te arrij për ta formuar lidhjen. Programimi i soketave mund te behet ne cilëndo gjuhe programuese. Serveri është program qe ofron disa shërbime ndërsa klienti kërkon shërbime. Soketet mund te përkufizohen si pike përfundimtare te lidhjes ne mes te dy kompjuterëve te identifikuar nga një IP adrese dhe port. Soketi është një ndërfaqe ne mes te aplikacionit dhe rrjetës(TCP). Soketi po ashtu mundëson dërgimin e te dhënave, marrjen e te dhënave dhe mbylljen e lidhjes. Porti është një komponentë e specifike softuerike e cila shërben si një pike fundore komunikuese në një sistem operativ hostues te kompjuterit. Një port është i asocuar me IP adresën e hostit si dhe tipin e protokollit qe përdoret për komunikim.

Për dizajnimin klient-server është përdorur serveri Visual Studio Code 2020 1.52.1. Sistemi operativ ku është bërë testimi i programeve është Windows 10. Versioni I python është Python 3.9.0 64-bit. Në këtë projekt janë krijuar disa metoda me te cilat komunikojnë klienti dhe serveri.

1. Protokolli FIEK-TCP

TCP do te thotë Transmission Control Protocol ose protokoll për kontrollin e transmetimit. Protokolli TCP transferon të dhënat në formën e rrjedhës së bashkuar të bajtëve. TCP grupon bajtët në formën e segmenteve TCP dhe më pas i kalon ato në shtresën IP(Internet Protocol) për transmetim në destinacion. Siguron integritetin e të dhënave që barten përmes një rrjeti. TCP përdoret për të transmetuar të dhëna nga protokollet e nivelit të lartë që u duhen të gjitha të dhënat për të arritur (Fortinet, 2021). Shpejtësia e protokollit TCP është e vogël, po besueshmëria e këtij protokolli është e madhe. Për dizajnimin klient-server kemi përdorur protokollin TCP, TCP Klienti dhe TCP Serveri ku tek TCP serveri kemi vendosur metodat te cilat mund te thirren nga klienti sipas dëshirës.

1.2 TCP Klienti

Tek TCP klienti pjesa e parë e kodit që ekzekutohet është kjo pjesë (Figura 1). Kjo pjesë si fillim ofron mundësinë e ndryshimit të IP adresës apo portit të klientit apo ta lëmë të nënkuptuar (IP:127.0.0.1, Porti: 14000), pastaj krijohet soketi i klientit dhe lidhet me server nëse është e mundur nëse jo me exception handling të definuar në kod na paraqitet teksti duket na lajmëruar që krijimi i soketit apo lidhja me server dështoi.

```
55  print("\nShëno adresen (by default: 127.0.0.1): ")
56  EmriIServerit=input()
57  print("Shëno portin(by default 14000): ")
58  PortiIServerit=input()
59
60  #Si host server i nënkuptueshem përdoret 'localhost'(127.0.0.1)
61  if EmriIServerit=="" or EmriIServerit=="localhost":
62      EmriIServerit="localhost"
63
64  #Si port i nënkuptueshëm përdoret 14000
65  if PortiIServerit=="" or PortiIServerit=="14000":
66      PortiIServerit=14000
67
68  adresa=(EmriIServerit,PortiIServerit)
69
70  #Provon të lidhet me serverin e shënuar ose me serverin e nënkuptuar
71  try:
72      #Krijimi i soketit të klientit
73      #Parametri i dytë i soketit tregon që kemi të bëjmë më TCP soket
74      clientS=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
75      #Vendoset lidhja ndërmjet klientit dhe serverit
76      clientS.connect(adresa)
77  except socket.error as mesazhi:
78      print("Krijimi i soketit dështoi.")
79      NDRYSHO()
```

Figure 1 TCP klienti

Në këtë pjese të kodit është shkruar funksioni NDRYSHO i cili mundëson ndryshimin e IP adresës apo portit te serverit(Figura 2). Tek try krijojmë soketin për klientin pastaj me connect lidhet me serverin. Në variablen kërkesa vendoset kërkesa nga klienti dhe shndërrohet në shkronja të mëdha me metodën upper().

```
def NDRYSHO():
    print("Shëno IP adresen: ")
    EmriIServerit=input()
    print("Shëno portin: ")
    PortiIServerit=int(input())
    adresa=(EmriIServerit,PortiIServerit)
    try:
        #Krijimi i soketit te klientit
        #Parametri i dyte i soketit tregon qe kemi te bejme me TCP soket
        clientS=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        #Vendoset lidhja ndermjet klientit dhe serverit
        clientS.connect(adresa)
        kerkesa=input("Operacioni (IP, NRPORTIT, NUMERO, ANASJELLTAS, PALINDROM, KOHA, LOJA, GCF, KONVERTO, FAKTORIELI, CEASAR, FIBONACCI)?: ")
        kerkesa=kerkesa.upper()
        while True:
            #Behet enkodimi i kerkeses se marre dhe dergohet tek TCP serveri
            clientS.sendall(str.encode(kerkesa))
            #Pranohet pergjigjja nga serveri dekodohet dhe madhesia me e madhe qe mund te mirret eshte 128 bajt
            data=clientS.recv(128)
            print(data.decode("utf-8"))
            kerkesa=input("Kërkësa juaj: ")
            if(kerkesa=="NDRYSHO"):
                NDRYSHO()
            elif(kerkesa=="NDALO"):
                clientS.close()
    except socket.error as mesazhi:
        print("Krijimi i soketit dështoi.Provoni përseri për një qasje tjetër ")
```

Figure 2 TCP klienti funksioni NDRYSHO()

Në këtë pjese të kodit merret kërkesa nga klienti dhe testohet, nëse ajo është NDALO, NDRYSHO apo nëse është ndonjë kërkesë tjetër dërgohet tek serveri për të marrë përgjigjën e kërkesës(Figura 3). Edhe këtu është përdorur exception handling nëse klienti nuk mund të lidhet me serverin na paraqitet teksti përkatës për të na lajmëruar.

```
#Kërkesa nga klienti testohet se cfarë kerkohet, me kushtëzimet në if else
try:
    kerkesa=input("Operacioni (IP, NRPORTIT, NUMERO, ANASJELLTAS, PALINDROM, KOHA, LOJA, GCF, KONVERTO, FAKTORIELI, CEASAR, FIBONACCI)?: ")
    #Të gjitha kërkesat të marrura si inpute i kthen në shkronja të mëdha
    kerkesa=kerkesa.upper()
    i=1
    while (i==1):
        if ((kerkesa=='NDALO')or(kerkesa=='')):
            i=2
            #Soketi mbyllet
            clientS.close()
            print("Lidhja me serverin është mbyllur, falemniderit për përdorimin")
        elif (kerkesa=="NDRYSHO"):
            #Thirret funksioni NDRYSHO()
            NDRYSHO()
        else:
            #Bëhet enkodimi i kërkesës së marrë dhe dërgohet tek TCP serveri
            clientS.sendall(str.encode(kerkesa)) ###
            #Pranohet pergjigjja nga serveri dekodohet dhe madhësia me e mesazhit që mund të merret është 128 bajt
            data=clientS.recv(128).decode()
            print(data)
            print("-----------------------------------------------------------------------------------------------------------")
            kerkesa=input("Operacioni (IP, NRPORTIT, NUMERO, ANASJELLTAS, PALINDROM, KOHA, LOJA, GCF, KONVERTO, FAKTORIELI, CEASAR, FIBONACCI)?: ")
            kerkesa=kerkesa.upper()
except:
    print("Nuk mund të arrihet qasja")
```

Figure 3 TCP klienti

1.3 TCP Serveri

Në fillim të programit deklarohet emri i serverit dhe porti pastaj krijohet soketi i serveri. Me except handling shikojmë nëse serveri mund të lidhet me ndonjë klient(Figura 4).

```
5   #IP adresa  IPv4
6   EmriIServerit = '127.0.0.1'
7   #Porti 12000 ndahet per soketin e serverit
8   PortiIServerit= 14000
9
10  adresa=(EmriIServerit,PortiIServerit)
11  #Krijimi i soketit te servetit sipas TCP protokollit
12  #socket.AF_INET pranon vetëm IP versioni 4
13  #socket.SOCK_STREAM mesazhet që i dergojmë shkojnë si të dhëna të kthyera në bajta
14  serverS = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
15
16  #Provon nëse serveri mund të lidhet me soketin e klientit
17  try:
18      #Lidhja e portit të serverit dhe ip adreses me soketin e krijuar
19      serverS.bind(adresa)
20  except socket.error:
21      print("Lidhja me klientin nuk është arritur!")
22      sys.exit()   #importojm librarin per sys
23
24  print('Serveri është startuar në localhost në portin:  ' + str(PortiIServerit))
25  #Serveri mund te pranoj 10 kerkesa
26  serverS.listen(10)
27  print('Server është duke punuar dhe është duke pritur për ndonjë kërkesë!')
```

Figure 4 TCP serveri

Krijimi i thread funksionit i cili pranon kërkesën nga klienti dhe me pas e lexon se cila është kërkesa e shkruar nga klienti ne mënyre që ti kthehet përgjigjja klientit(Figura 5).

```
159 def Thread(connection):
160     while True:
161         try:
162             #Kërkesa e klientit merret dhe dekodohet
163             #Kërkesa e pranuar mund të jetë më së shumti 128 bajt
164             data=connection.recv(128).decode()
165         except socket.error:
166             print("Klienti me IP "+ adresa[0]+" u shkyq nga serveri!")
167             break
168 
169         print(" ")
170         #Të dhënat nga klienti kthehen në string
171         getdata=str(data)
172 
173         #rsplit() kthen një listë të vargjeve pasi thyen vargun e dhënë nga ana e djathtë me ndarës të specifikuar.
174         ListaEFjaleve=getdata.rsplit(" ")
175         ListaEFjaleve[0]=ListaEFjaleve[0].upper()
176         rreshti=""
177         GjatesiaListes=len(ListaEFjaleve)
178 
179         #Merret secila fjal pas kerkeses së shkruar
180         for fjalet in range(1, GjatesiaListes):
181             rreshti +=ListaEFjaleve[fjalet]
182             if(fjalet!=GjatesiaListes):
183                 rreshti+=" "
184 
185         #Nëse nuk ka të dhena ndalon
186         if not data:
187             break
188         #Nëse ka të dhëna hyrëse(nëse është shkruar kërkesa) do të vizitoj secilen loop deri sa të gjendet kërkesa e shënuar
189         elif(ListaEFjaleve[0]=="IP"):
190             data="IP Adresa e klientit është: " + IP()
191 
192         elif(ListaEFjaleve[0]=="NRPORTIT"):
193             data="Klienti është duke përdorur portin: " + NRPORTIT()
```

Figure 5 TCP klienti funksioni Thread

Kjo loop ekzekutohet çdo herë kur serveri lidhet me ndonjë klient po ashtu është krijuar edhe mundësia e lidhjes se disa klientëve në të njëjtën kohe në një server do të thotë është implementuar libraria thread(Figure 6). Thread është një sekuencë e udhëzimeve të tilla brenda një programi që mund të ekzekutohet në mënyrë të pavarur nga kodi tjetër (GeeksforGeeks, 2019).

```
from _thread import *
251 while True:
252     connection, addressaa=serverS.accept()
253     print(f"Serveri është lidhur me klientin në IP adresë %s në portin %s"% addressaa)
254     #Metoda start_new_thread mundëson në menyrë të shpejtë dhe efikase lidhjen me nje klient tjeter(me disa klient)
255     start_new_thread(Thread,(connection,))
256 serverS.close()
```

Figure 6 Implementimi i thread në TCP klienti

2. Protokolli FIEK-UDP

UDP do të thotë User Datagram Protocol ose protokolli i të dhënave të përdoruesit. Është protokoll jo i besueshëm dhe nuk ka lidhje. Kështu nuk ka nevojë të vendoset lidhje para transferimit të të dhënave. Është i shpejte për transferimin e të dhënave. Ky protokoll përdoret për shërbimet në kohë reale si lojërat kompjuterike, komunikimi zanor ose video, konferenca drejtpërdrejt. Edhe për protokollin UDP kemi dizajnuar klientin dhe serverin që komunikojnë me njeri-tjetrin.

2.1 UDP Klienti

I njëjti kod si tek TCP klienti është implementuar edhe tek UDP klienti vetëm dallon tek krijimi i soketes në vend të socket.SOCK_STREAM tek UDP vendosim socket.SOCK.DGRAM(Figure 7).

```
34          #Krijimi i soketit te klientit
35          #Parametri i dyte i soketit tregon qe kemi te bejme me UDP soket
36          clientS=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
```

Figure 7 UDP klienti krijimi i soketit

2.2 UDP Serveri

UDP server dallon nga TCP server në krijimin e soketit.

```
serverS = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

Po ashtu tek UDP serveri funksioni Thread pranon dy parametra vlerën hyrëse dhe adresën. Ne Figure 8 është paraqitur implementimi i thread ne këtë versionte protokollit.

```
235 while True:
236     data, adresaa=serverS.recvfrom(128)
237     print(IP() + " ka bere kerkesen -> "+data.decode("utf-8"))
238     start_new_thread(Thread,(data,adresaa))
239 serverS.close()
```

Figure 8 Implementimi i thread UDP serveri

3. Përshkrimi i kërkesave

Për dizajnimin klient server në protokollin TCP dhe UDP janë implementuar këto metoda:

- IP
- NRPORITIT
- NUMERO
- ANASJELLTAS
- PALINDROM
- KOHA
- LOJA
- GCF
- KONVERTO

Metodat shtesë:

- FAKTORIELI
- CEASAR
- FIBONACCI

3.1. Metoda IP

Kjo metodë përcakton dhe kthen IP adresën e klientit. Për implementimin e kësaj metode është krijuar një funksion me emrin IP i cili brenda përmban variablen përgjigjja. Funksioni str() përdoret për shndërrimin e tekstit brenda kllapave në string kështu variabla përgjigjja ka marrë stringun addressaa[0] dhe kjo përgjigje kthehet me return. Në figure 9 është paraqitur funksioni IP dhe pamja e kodit për identifikimin e kërkesës nëse është IP atëherë shfaqet tekstit pastaj adresa e klientit.

```
def IP():
    pergjigjja = str(addressaa[0])
    return pergjigjja
```

```
elif(ListaEFjaleve[0]=="IP"):
    data="IP Adresa e klientit është:" + IP()
```

Figure 9 Funksioni IP dhe procedimi i kërkesës

3.2. Metoda NRPORTIT

Kjo metodë përcakton dhe kthen portin e klientit. Për implementimin e kësaj metode është krijuar një funksion me emrin NRPORTIT i cili brenda përmban variablen përgjigjja. Funksioni str() përdoret për shndërrimin e tekstit brenda kllapave në string kështu variabla përgjigjja ka marrë stringun addressaa[1] dhe kjo përgjigje kthehet me return. Në figure 10 është paraqitur funksioni NRPORTIT pamja e kodit për identifikimin e kërkesës nëse është NRPORTIT atëherë shfaqet tekstit pastaj porti i klientit.

```
def NRPORTIT():
    pergjigjja =str(addressaa[1])
    return pergjigjja
```

```
elif(ListaEFjaleve[0]=="NRPORTIT"):
    data="Klienti është duke përdorur portin: " + NRPORTIT()
```

Figure 10 Funksioni NRPORTIT dhe procedimi i kërkesës

3.3. Metoda NUMERO

Funksioni NUMERO kthen numrin e zanoreve dhe bashkëtingëlloreve të fjalës së dhënë nga klienti. Ne figure 11 eshte paraqitur funksioni dhe procedimi i kërkesës nëse klienti e shkruan stringun NUMERO.

```
def NUMERO(tekst):
    zanore=0
    bashketingellore=0
    for i in tekst:
        if (i == 'a'or i == 'e'or i == 'i'or i == 'o'or i == 'u'or i=='A' or i=='E'or i=='I' or i=='O'or i=='U'):
            zanore=zanore+1
        else:
            bashketingellore=bashketingellore+1
    pergjigjja=str("Teksti ka "+str(zanore) +" zanore dhe " +str(bashketingellore)+" bashkëtingëllore.")
    return pergjigjja
```

```
elif (ListaEFjaleve[0]=="NUMERO"):
    data=str(NUMERO(rreshti))
```

Figure 11 Funksioni NUMERO dhe procedimi i kërkesës

3.4. Metoda ANASJELLTAS

Kjo metodë kthen tekstin e anasjelltë ose reversë. Në TCPserveri dhe UDPserveri është deklaruara funksioni ANASJELLTAS me parametër hyrës të emëruara tekst (figura 1). Në variabël ruhen shkronjat por nga fundi i tekstit tekst[::-1] do të thotë fillon nga fundi i stringut dhe vazhdon në të majte deri në 0. Edhe elif është iplementuar në TCPserveri dhe UDPserveri për të shikuar nëse kërkesa e marrë nga klienti është ANASJELLTAS nëse po atëherë kthehet vlera pasi thirret funksioni (figura 2).

```
def ANASJELLTAS(tekst):
    pergjigjja=tekst[::-1]
    return pergjigjja
```

```
elif(ListaEFjaleve[0] == "ANASJELLTAS"):
    data=str(ANASJELLTAS(rreshti))
```

Figure 12 Funksioni ANASJELLTAS dhe procedimi i kërkesës

3.5 Metoda PALINDROM

Per implementimin e metodës PALINDROM kemi krijuar nje funksion i cili ka nje arameter hyrës dhe kthen nje tekset qe tregon nëse teksti ne hyrje eshte palindrom ose jo(Figure 13) .

```
def PALINDROM(tekst):
    w=""
    for i in tekst:
        w=i+w

    if(tekst==w):
        return "Teksti i dhënë është palindrome"
    else:
        return "Teksti nuk është palindrome"
```

```
elif(ListaEFjaleve[0] == "PALINDROM"):
    data=str(PALINDROM(ListaEFjaleve[1]))
```

Figure 13 Funksioni PALINDROM dhe procedimi i kërkesës

3.6. Metoda KOHA

Funksioni KOHA e kthen kohën aktuale. Për implementimin e metodës koha e nevojshme ka qenë edhe importimi nga libraria time i gmtime në server.

```
def KOHA():
    from time import gmtime, strftime
    pergjigjja = strftime("%d.%m.%Y %I:%M:%S %p")
    return pergjigjja
```

```
elif(ListaEFjaleve[0] == "KOHA"):
    data=KOHA()
```

Figure 14 Funksioni KOHA dhe procedimi i kërkesës

3.7. Metoda LOJA

Funksioni kthen 5 numra të rastësishëm në intervalin caktuar(1,35). Për implementimin e kësaj metode e nevojshme ka qenë importimi i librarisë random në fillim të programit.

```
def LOJA():
    NumerRandom = ""
    for x in range(1,6):
        #radint() kthen një numër të plotë
        randomNu = random.randint(1,35)
        NumerRandom += str(randomNu) + " "
    return NumerRandom
```

```
elif(ListaEFjaleve[0] == "LOJA"):
    data=LOJA()+"pra 5 numra të rastësishëm nga 35"
```

Figure 15 Funksioni LOJA dhe procedimi i kërkesës

3.8. Metoda GCF

Funksioni GCF kthen faktorin më të madh të përbashkët në mes dy numra. Është implementuar edhe exception handling tek procedimi i kërkesës ne rastet kur hyrjet e kërkesës klienti i shënon gabim. Modelimi i metodës GCF është prezantuar në vijim.

```
def GCF(numri1,numri2):
    if numri1%numri2==0:
        return numri2
    return GCF(numri2,numri1%numri2)
```

```
elif(ListaEFjaleve[0]== "GCF"):
    try:
        rreshti = int(ListaEFjaleve[1])
        rreshti1= int(ListaEFjaleve[2])
        data=str(GCF(rreshti,rreshti1))
    except Exception:
        data = "Shtypni komanden si të tillë GCF Numëri1 Numëri2"
```

Figure 16 Funksioni GCF dhe procedimi i kërkesës

3.9 Metoda KONVERTO

Funksioni Konverto kthen si rezultat konvertimin e opcioneve varësisht opcionit të zgjedhur. Funksioni pranon dy parametra dhe kthen vetëm një parametër(numër) që merr vetëm dy numra pas pikës dhjetore. Ky funksion bën konvertimin e njësive të gjatësis varësisht prej opsionit që zgjedh klienti. Opsionet janë: cmtoinch, inchnecm, kmnemiles, milenekm. Modelimi i metodës konvert është prezantuar në vijim.

```
def KONVERTO(zgjedhja, vlera):
    if(zgjedhja=="CMTOINCH"):
        #Metoda format kthen vlerën e numrit të përcaktuar duke marrë parasysh .2f
        pergjigjja=format((vlera / 2.54),".2f")
    elif(zgjedhja=="INCHNECM"):
        pergjigjja = format((vlera * 2.54),".2f")
    elif(zgjedhja=="KMNEMILES"):
        conv_fac=0.621371
        pergjigjja= format((conv_fac * 2.54),".2f")
    elif(zgjedhja=="MILENEKM"):
        conv_fac=1.60934
        pergjigjja= format((conv_fac * 2.54),".2f")
    else:
        pergjigjja = "Komand jo valide"
    return pergjigjja
```

```
elif(ListaEFjaleve[0] == "KONVERTO"):
    try:
        number = float(ListaEFjaleve[2])
        data="Vlera e fituar është : " + str(KONVERTO(ListaEFjaleve[1], number))
    except Exception:
        data="Gabim jepni njëhere opsionin pastaj numrin"
```

Figure 17 Funksioni KONVERTO dhe procedimi i kërkesës

3.10 Metoda FAKTORIELI

Metoda faktorieli është njëra nga metodat shtesë të punuara e cila është e thjeshtë për implementim. Kjo metode merr një numrin, kthen faktorielin e atij numri dhe përdoret rekurzioni brenda metodës.

```
def FAKTORIELI(n):
    if n==0:
        return 1
    else:
        return n* FAKTORIELI(n-1)
```

```
elif (ListaEFjaleve[0]=="FAKTORIELI"):
    try:
        number=int(ListaEFjaleve[1])
        data="Faktorieli i numrit që keni dhënë është: "+ str(FAKTORIELI(number))
    except Exception:
        data="Duhet te shenoni numrin!"
```

Figure 18 Funksioni FAKTORIELI dhe procedimi i kërkesës

3.11 Metoda CEASAR

Metoda ceasar është njëra nga metodat shtesë të punuara e cila enkripton ose dekripton tekstin në bazë të opsionit që klienti shkruan. Nëse opsioni është zgjedhur është enkripto fjalia e dhëne do të kthehet e enkriptuar për secilën shkronje të zhvendosur për 5 shkronja tjera të radhës shembull A->F, B->G etj. Ndërsa për dekriptim është e njëjta procedure vetëm se këtu zhvendosja bëhet për 5 shkronja mbrapa. Nëse si vlere hyrëse ka numër atëherë kthehet përsëri i njëjti numër nuk ndryshohet .

```python
def CEASAR(fjalia,opsioni):
    opsioni=opsioni.upper()
    fjalia=fjalia.upper()
    if((opsioni=="ENKRIPTO")):
        tekstiEnkriptuar=""
        for shkronja in fjalia:
            #Funksioni isalpha () kthehet e vërtetë(True) nëse të gjithë karakteret janë shkronja alfabeti (a-z).
            if(shkronja.isalpha()):
                #Funksioni ord() kthen vlerën e shkronjës që ka në bazë të tabelës ASCII
                #Funksioni chr() kthen numrin në shkronjë duke u bazuar në tabelëen e ASCII
                #Funksioni str() kthen tekstin në string
                tekstiEnkriptuar+=str(chr(((ord(shkronja)-65+5)%26)+65))
            else:
                tekstiEnkriptuar+=shkronja
        return tekstiEnkriptuar
    elif(opsioni=="DEKRIPTO"):
        tekstiDekriptuar=""
        for shkronja in fjalia:
            if(shkronja.isalpha()):
                tekstiDekriptuar+=str(chr(((ord(shkronja)-65-5)%26)+65))
            else:
                tekstiDekriptuar+=shkronja
        return tekstiDekriptuar
```

```python
elif(ListaEFjaleve[0]=="CEASAR"):
    fjaliaCeasarit=""
    for i in range(2,len(ListaEFjaleve)):
        fjaliaCeasarit+=ListaEFjaleve[i]
        if(fjalet!=i):
            fjaliaCeasarit+=" "
    if(ListaEFjaleve[1]=="ENKRIPTO"):
        data="Teksti i enkriptuar me algoritmin e Ceasar është: "+CEASAR(fjaliaCeasarit,"ENKRIPTO")
    elif(ListaEFjaleve[1]=="DEKRIPTO"):
        data="Teksti i dekriptuar me algoritmin e Ceasar është: "+CEASAR(fjaliaCeasarit,"DEKRIPTO")
    else:
        data="Keni dhënë kërkesë të gabuar"
```

Figure 19 Funksioni CEASAR dhe procedimi i kërkesës

3.12 Metoda FIBONACCI

Edhe metoda fibonacci është njëra nga metodat shtese e cila është relativisht e thjeshte për implementim. Ky funksion pranon një parametër në bazë të numrit që shtyp klienti dhe kthen fibonaccin e atij numri. Nëse numri është me i vogël se 0 shfaqet se numri duhet te jete pozitiv dhe po ashtu përdoret rekusioni.

```python
def FIBONACCI(n):
        if n < 0:
            print("Numri duhet të jetë pozitiv")
        elif n == 0:
            return 0
        elif n == 1 or n == 2:
            return 1
        else:
            return FIBONACCI(n-1) + FIBONACCI(n-2)
```

```python
elif (ListaEFjaleve[0]=="FIBONACCI"):
    try:
        number=int(ListaEFjaleve[1])
        data="Fibonacci i numrit që keni dhënë është: "+ str(FIBONACCI(number))
    except Exception:
        data="Duhet te shenoni numrin!"
```

Figure 20 Funksioni FIBONACCI dhe procedimi i kërkesës

Testime

Në këtë sektor do të prezantojmë rezutatatet e arritura dhe testimet e bëra. Fillimisht në figurën 21, janë prezantuar të dhënat që shfaqen pas ekzekutimit të TCP Klient dhe TCP Server në terminal.

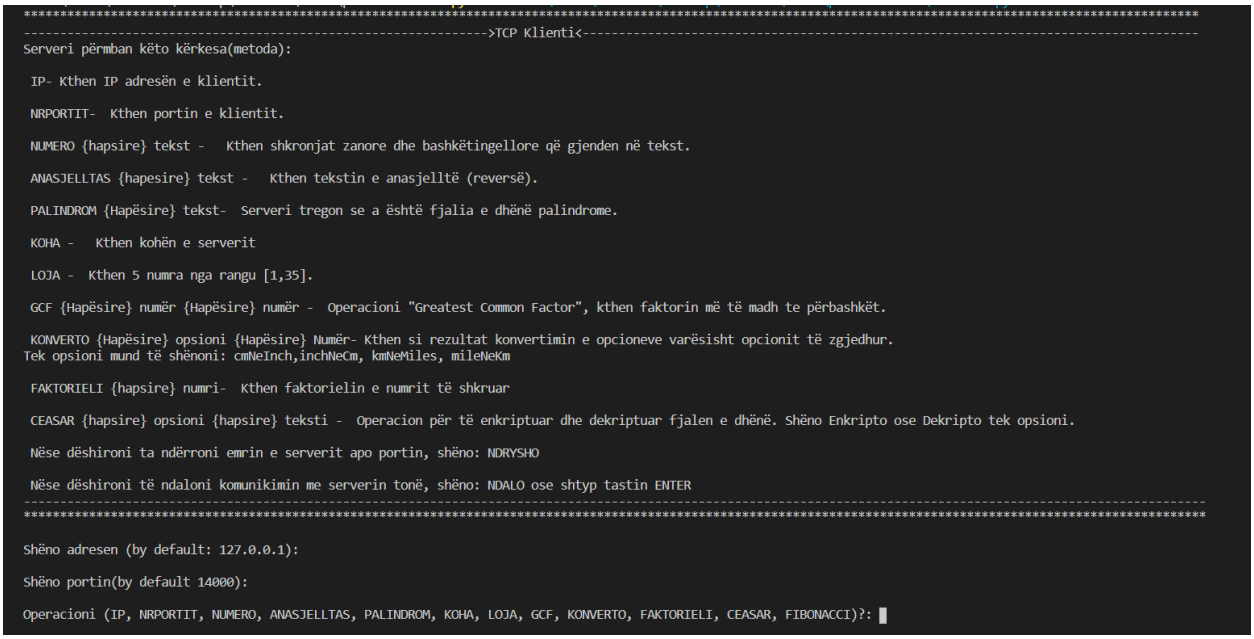

Figure 21 TCP klienti terminali

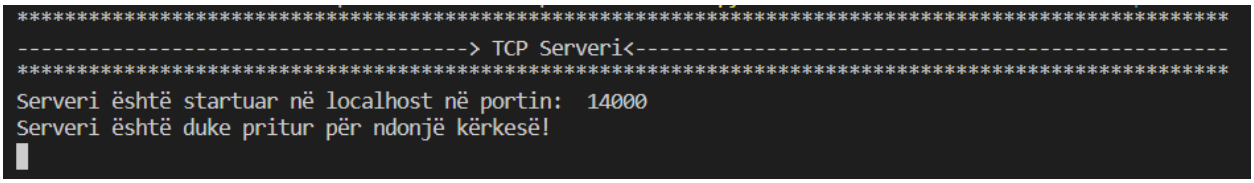

Figure 22 TCP Serveri terminali

Në vijim janë paraqitur rezultatet e testuara për të gjitha kërkesat. Në figure 23 janë paraqitur rezultatet e fituara në TCP Klient ndërsa në figure 24 rezultatet e fituara në TCP Server.

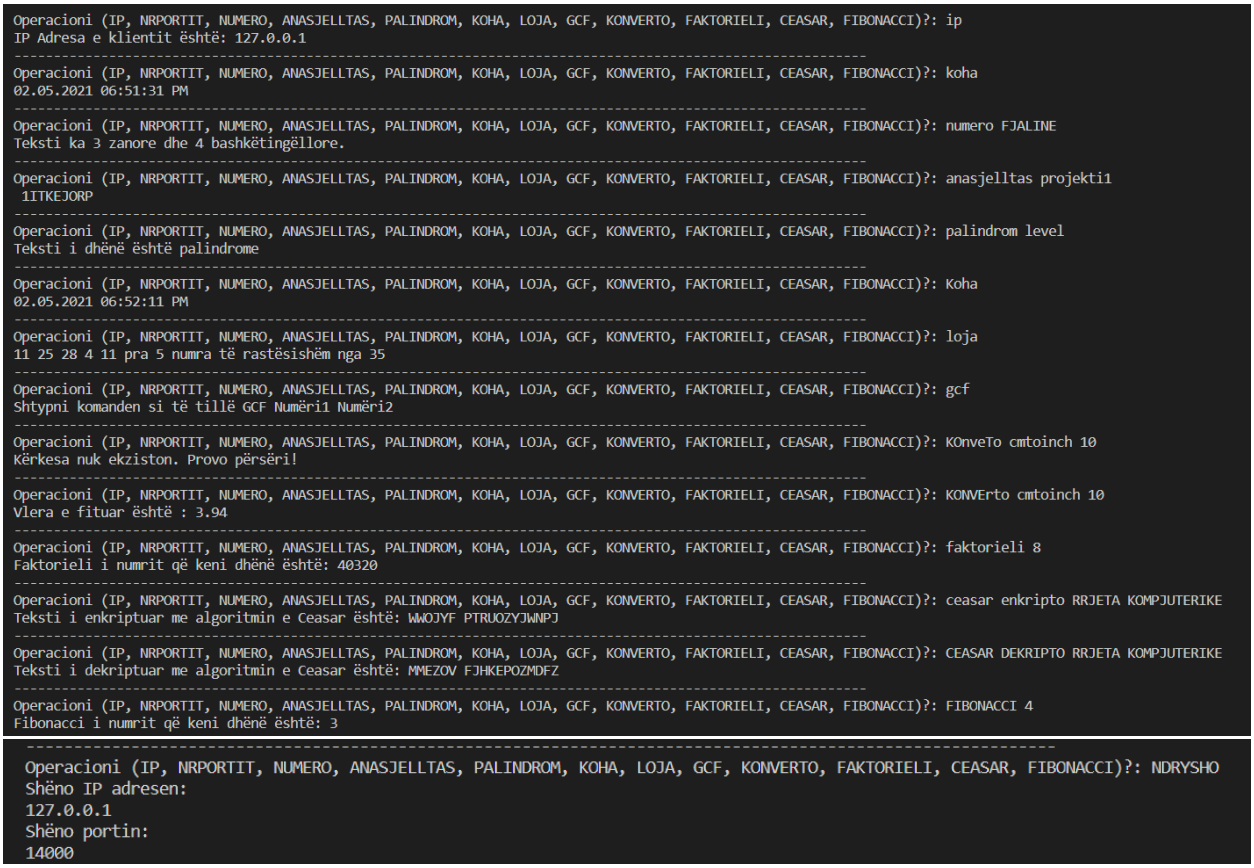

Figure 23 TCP klienti terminali

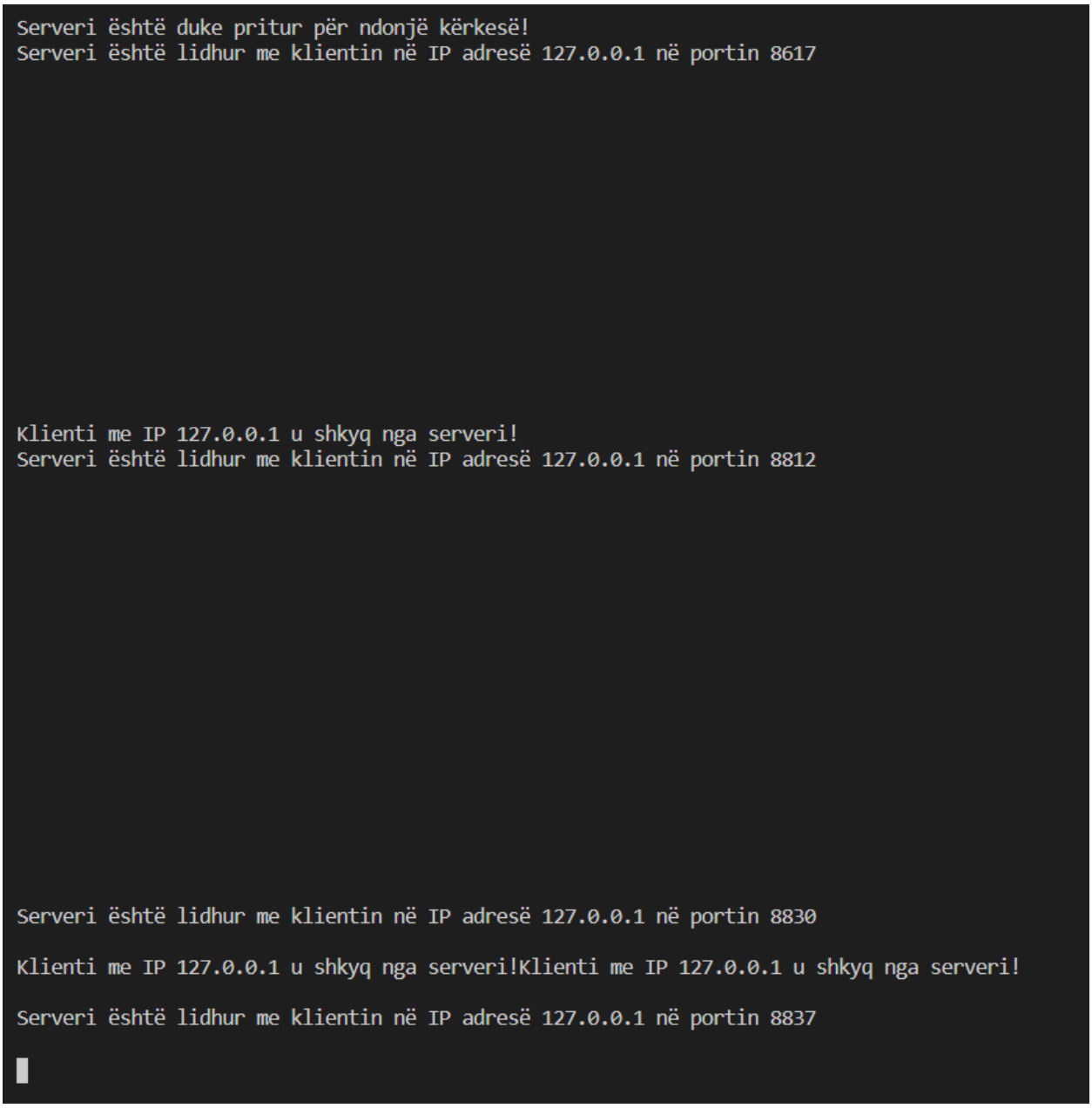

Figure 24 TCP serveri terminali

Në figure 25,26 është paraqitur testimi i multithreads në protokollon TCP dhe UDP, komunikimi i tre klientëve me një server.

Protokolli TCP

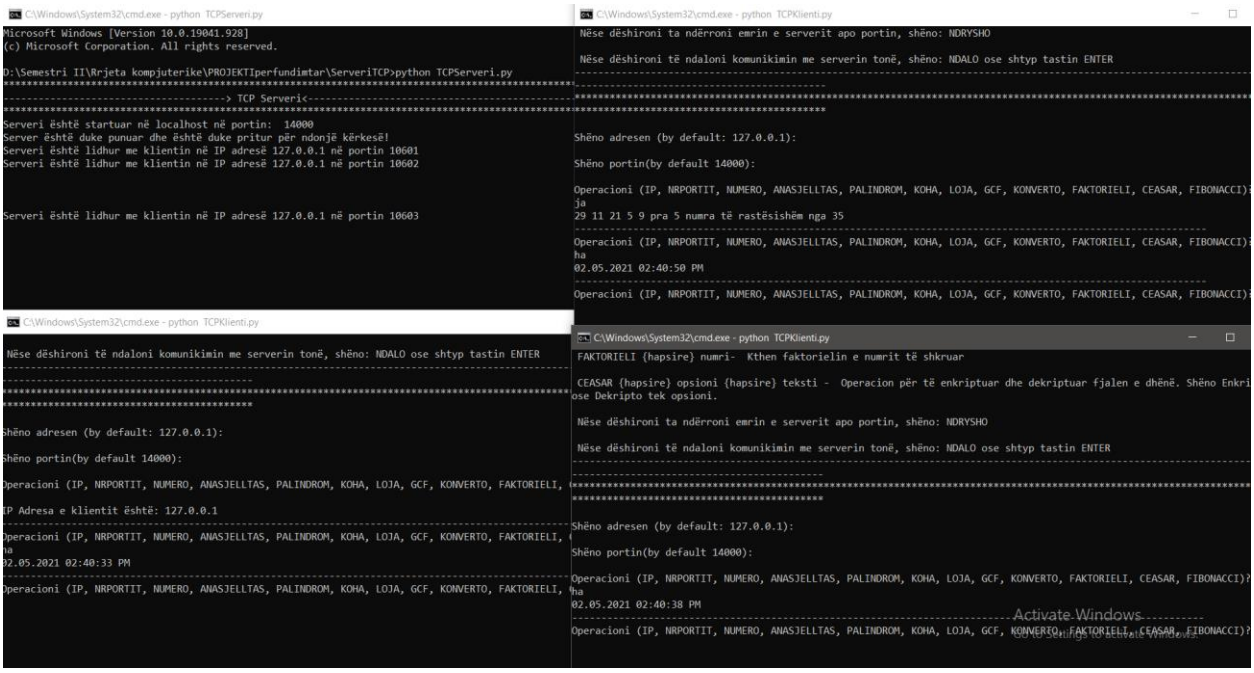

Figure 25 TCP Multithread

Protokolli UDP

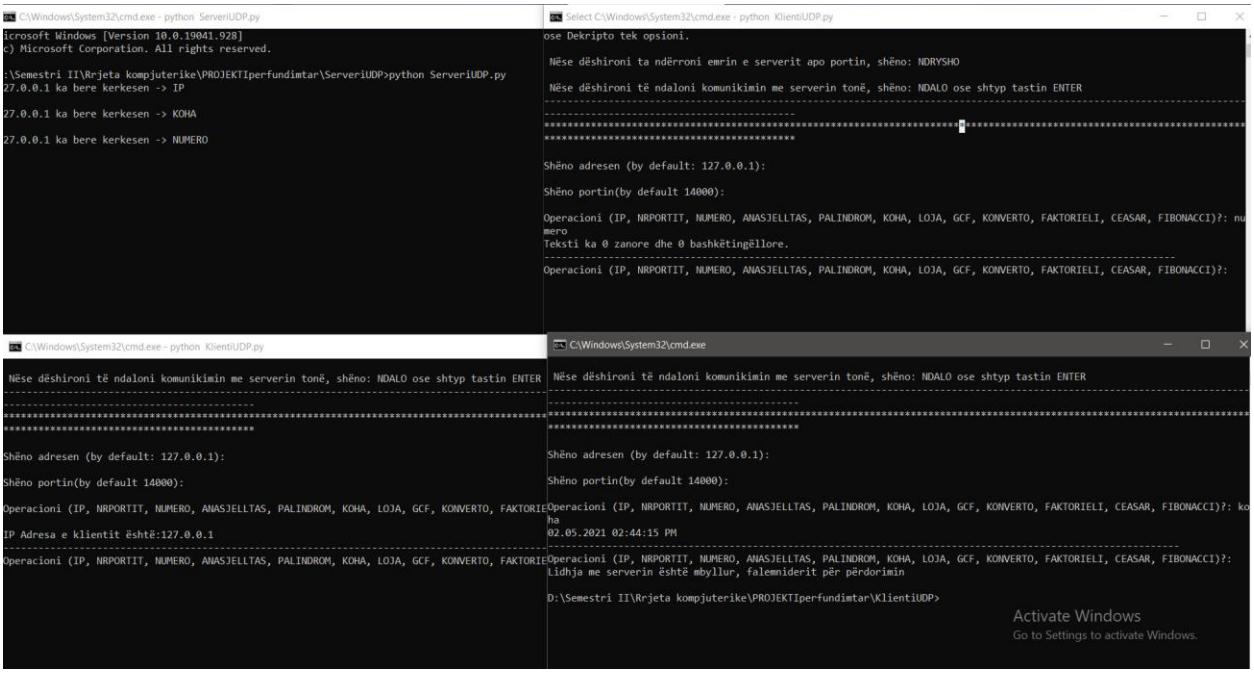

Figure 26 UDP Multithread

Përfundime

Me përfundimin e këtij projekti i kam përvetësuar më shumë njohuritë rreth protokolleve TCP dhe UDP, soketat dhe programimin e tyre me TCP dhe UDP po ashtu edhe avancimin e njohurive te programimit ne gjuhen programuese python. Njohuritë janë përvetësuar me hulumtimin për plotësimin e kërkesave te kërkuara nga projekti. Raporti përmban të gjitha kërkesat e zgjidhura si dhe kërkesat shtesë prej të cilave tre kërkesa i kam shtuar, të gjitha kërkesat funksionojnë në mënyrë të duhur dhe kthejnë rezultatet e kërkuara.

Hulumtimet për literaturë

- PositronX (https://www.positronx.io/create-socket-server-with-multiple-clients-in-python/)
- GreeksforGreeks (https://www.geeksforgeeks.org/python-program-check-string-palindrome-not/)
- Includehelp (https://www.includehelp.com/python/program-to-convert-centimeter-to-inches.aspx)
- Stackoverflow (https://stackoverflow.com/questions/415511/how-to-get-the-current-time-in-python)
- Tutorialspoint (https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_caesar_cipher.htm)
- Python.plainenglish (https://python.plainenglish.io/caesar-cipher-in-python-cc339686725b)
- Stackoverflow (https://stackoverflow.com/questions/27218415/python-socket-programming-with-multiple-threads)
- Javapoint (https://www.javatpoint.com/python-functions)
- Learnpython (https://www.learnpython.org/en/Functions)
- Programiz (https://www.programiz.com/python-programming)
- Stackoverflow (https://stackoverflow.com/questions/tagged/python)
- Codementor (https://www.codementor.io/community/topic/python)
- Pitt (https://www.pitt.edu/~naraehan/python3/user_defined_functions.html)

Referencat

[1]. *Fortinet*. (2021). Retrieved from Fortinet.com: https://www.fortinet.com/resources/cyberglossary/tcp-ip#form

[2]. *GeeksforGeeks*. (2019, November 29). Retrieved from geeksforgeeks.org: https://www.geeksforgeeks.org/socket-programming-multi-threading-python/