**Universiteti i Prishtinës**

Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike

Lënda: Rrjetat Kompjuterike – 2017



Projekti 1:

***Dizajnimi Klient-Server duke përdorur Socketët (ang.Sockets)***

***Punoi: Pranoi:***

***Argjenda Dervisholli Msc. Edon Mustafa***

***24.04.2017, Prishtinë***

**Përmbajtja**

1. **Hyrje**
2. **Protokoli FIEK**
   1. **Përshkrimi i përgjithshëm**
   2. **Specifikimi i kërkesave**
3. **Programi Klient-Server**
   1. **Programi FIEK-TCP**
   2. **Programi FIEK-UDP**
4. **Përshrimi i kodit**
   1. **Përshkrimi i serverit UDP**
   2. **Përshkrimi i serverit TCP**
5. **Përshkrimi i kërkesave**
   1. **Kërkesa IP**
   2. **Kërkesa PORT**
   3. **Kërkesa ZANORE**
   4. **Kërkesa PRINTO**
   5. **Kërkesa HOST**
   6. **Kërkesa TIME**
   7. **Kërkesa KENO**
   8. **Kërkesa FAKTORIEL**
   9. **Kërkesa KONVE**
   10. **Kërkesa AREA**
   11. **Kërkesa HARMONIK**

**1.1 Hyrje**

**Objektivat edukative të këtij projekti janë që ti përforcoj konceptet dhe protokollet që lidhen me arkitekturën klient/server dhe programimin me socket me protokollet TCP dhe UDP. Qëllimi I projektit është dizajnimi, implementimi dhe testimi i programit klient dhe server sipas specifikave të protokollit FIEK.**

**2. Protokoli FIEK**

**2.1 Përshkrimi i përgjithshëm**

Projekti përbëhet nga sistemi programit të klientit dhe serverit të cilët komunikojnë përms FIEK Protokollit.

Komunikimi ndermjet klientit dhe serverit përmes socket-ave të cilat mundesojne komunikim e tyre në portin default 9000 ose një port të vecantë të cilin mund ta caktojë klienti.

FIEK Protokolli është një protokoll i thjeshtë i komunikimit i cili per bazë ka fjalen kyqe në fillim të kërkesës e cila percakton metodën të cilën do ta kthej server. Ajo ka dy versione:

1. FIEK-TCP
2. FIEK-UDP
   1. **Specifikimi i kërkesave**

Ky protokoll ju përgjigjet vetëm këtyre kerkesave: IP, PORT, ZANORE, PRINTO, HOST, TIME, KENO, FALTORIEL, AREA, HARMONIK, të cilat dërgohen nga klienti në server dhe për të cilat serveri kthen nga një përgjigje të vecantë, klienti së bashku me kërkesen mund të dërgojë edhe input varesisht nga kerkesa.

**2. Programi Klient-Server**

* 1. **Programi FIEK-TCP**

Së pari vendosët një lidhje(segment) në mes të klientit dhe serverit përmes socket-ave në portin e caktuar(default 9000). Pastaj klienti përmes dërgon kërkesën tek server, i cili nesër kërkesa është valide i përgjigjet kërkeses specifike të klientit dhe me pas mbyll lidhjen me klientin. Serveri është në gjendje të lidhet me me shumë kliente në të njejtën kohë dhe ti kthej përgjigje klientëve në mënyrë paralele.

* 1. **Programi FIEK-UDP**

Në versionin UDP të prokollit FIEK komunikimi realizohet vetëm përmes UDP datagram-ve, ku me c’rast nuk krijohet një lidhje(segment) mes klientit dhe serverit por komunikimi fillon drejt. Klienti dergon kërkesën tek server përmes një UDP datagram. Pasi që kërkesa të validohet në server ajo poashtu kthehet tek klienti përmes UDP datagram-it. Protokolli FIEK limiton që klienti të dergon vetem një kërkesë për datagram.

**4. Përshrimi i kodit**

**4.1 Pershkrimi i Serverit UDP**

Serveri është console aplikacion që lidhet me klient përmes socketave dhe komunikom me datagram-a. Serveri dallon llojet e kërkesave që vijnë nga klienti dhe për secilin lloj ka përgjigje të veqantë. Serveri është aplikacion që gjendet në unazë të pafundme gjë që nuk e lejon të ndalet, pra gjat tërë kohës është aktiv. Për një datagram serveri mund të dërgoj vetëm një datagram.

KODI:

from socket import\*

import random

import threading

from datetime import datetime

from random import randint

ip='10.10.7.251'

port=9000

host='localhost'

serverSocket=socket(AF\_INET,SOCK\_DGRAM)

serverSocket.bind(('',12001))

def recieveData(s,conn):

data=conn.recv(128)

print(conn,data)

return data;

def main():

s=socket(AF\_INET,SOCK\_STREAM)

s.bind((host,port))

s.listen(10)

print("Server eshte duke punuar.... \n")

conn, addr=s.accept()

while 1:

data = conn.recv(128)

if not data:

break

toBeReturned=str(if\_else\_func(data.decode('ASCII')))

conn.send(toBeReturned.encode('ASCII'))

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n-1)

def gjej\_zanoret(germa):

if (germa == 'a'

or germa =='A'

or germa == 'e'

or germa == 'E'

or germa == 'i'

or germa == 'I'

or germa == 'o'

or germa == 'O'

or germa == 'u'

or germa == 'U'

or germa == 'y'

or germa == 'Y'):

return True

def numero\_zanoret(fjala):

nr\_zanoreve = 0

for i in fjala:

rezultati = gjej\_zanoret(i)

if rezultati == True:

nr\_zanoreve += 1

return nr\_zanoreve

def area(width,height):

return width\*height

def convert(opsioni=None, numer=None):

if opsioni == "FahToCel":

return(numer - 32.0) \* (5.0/9.0)

elif opsioni == "CelToFah":

return(numer \* (9.0/5.0)) + 32.0

elif opsioni == "FahToKel":

return((numer - 32.0) \* (5.0/9.0)) + 273.15

elif opsioni == "KelToFah":

return((numer-273.15)\*(5.0/9.0))+32.0

elif opsioni == "CelToKel":

return(numer + 273.15)

elif opsioni == "KelToCel":

return(numer - 273.15)

elif opsioni == "KilToPou":

return(numer \* 2.2)

elif opsioni == "PouToKil":

return(numer / 2.2)

def keno():

rezultati = ""

i=0

while(i<20):

rezultati = rezultati + str(randint(1,80)) + ","

i=i+1

return rezultati

def harmonik(n, r=1):

total=0.0

for i in range(1,n+1):

total+=1.0/(i\*\*r)

return total

def if\_else\_func(message):

if message=='IP': return("IP addressa eshte =")+ip

if message=='PORTI': return port

if message=='KOHA': return datetime.now().strftime('%Y-%m-%d %H:%M:%S')

if message=='HOST': return host

if message[:6] == 'PRINTO': return message[7:]

if message[:9] == 'FAKTORIEL': return factorial(int(message[10:]))

if message[:6] == 'ZANORE': return numero\_zanoret(message[8:])

if message == 'KENO': return keno()

if message[:8] == 'KONVERTO': return convert(message[9:17], int(message[18:]))

if message[:4]=='AREA': return area(int(message[5:6]), int(message[7:8]))

if message[:8]=='HARMONIK': return harmonik(int(message[9:10]))

return "Komand e panjohur. Shkruani edhe nje here kerkesen."

main()

**KODI PER KLIENT**

import socket

import time

ip="127.0.0.1"

port=12001

clientSocket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

clientSocket.settimeout(1)

while 10:

message=input("Shkruani Kërkesën :")

socketClient.send(message.encode('ASCII'))

if\_else\_func=socketClient.recv(128)

print("Pergjigja: " + if\_else\_func.decode('ASCII'))

**4.1 Pershkrimi i Serverit TCP**

Serveri dallon llojet e kërkesave që vijnë nga klienti dhe për secilin lloj ka përgjigje të veqantë. Serveri është aplikacion që gjendet në unazë të pafundme gjë që nuk e lejon të ndalet, pra gjat tërë kohës është aktiv. Pas seciles përgjigje mbyllet soketi.

KODI:

from socket import\*

from datetime import datetime

from random import randint

ip='10.10.7.251'

port=9000

host='localhost'

def recieveData(s,conn):

data=conn.recv(128)

print(conn,data)

return data;

def broadcastData(s,conn,data):

conn.sendall(data)

print("data sent all clients \n")

def main():

s=socket(AF\_INET,SOCK\_STREAM)

s.bind((host,port))

s.listen(10)

print("Server eshte duke punuar.... \n")

conn, addr=s.accept()

while 1:

data = conn.recv(128)

if not data:

break

toBeReturned=str(if\_else\_func(data.decode('ASCII')))

conn.send(toBeReturned.encode('ASCII'))

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n-1)

def gjej\_zanoret(germa):

if (germa == 'a'

or germa =='A'

or germa == 'e'

or germa == 'E'

or germa == 'i'

or germa == 'I'

or germa == 'o'

or germa == 'O'

or germa == 'u'

or germa == 'U'

or germa == 'y'

or germa == 'Y'):

return True

def numero\_zanoret(fjala):

nr\_zanoreve = 0

for i in fjala:

rezultati = gjej\_zanoret(i)

if rezultati == True:

nr\_zanoreve += 1

return nr\_zanoreve

def area(width,height):

return width\*height

def convert(opsioni=None, numer=None):

if opsioni == "FahToCel":

return(numer - 32.0) \* (5.0/9.0)

elif opsioni == "CelToFah":

return(numer \* (9.0/5.0)) + 32.0

elif opsioni == "FahToKel":

return((numer - 32.0) \* (5.0/9.0)) + 273.15

elif opsioni == "KelToFah":

return((numer-273.15)\*(5.0/9.0))+32.0

elif opsioni == "CelToKel":

return(numer + 273.15)

elif opsioni == "KelToCel":

return(numer - 273.15)

elif opsioni == "KilToPou":

return(numer \* 2.2)

elif opsioni == "PouToKil":

return(numer / 2.2)

def keno():

rezultati = ""

i=0

while(i<20):

rezultati = rezultati + str(randint(1,80)) + ","

i=i+1

return rezultati

def harmonik(n, r=1):

total=0.0

for i in range(1,n+1):

total+=1.0/(i\*\*r)

return total

def if\_else\_func(message):

if message=='IP': return("IP addressa eshte =")+ip

if message=='PORTI': return port

if message=='KOHA': return datetime.now().strftime('%Y-%m-%d %H:%M:%S')

if message=='HOST': return host

if message[:6] == 'PRINTO': return message[7:]

if message[:9] == 'FAKTORIEL': return factorial(int(message[10:]))

if message[:6] == 'ZANORE': return numero\_zanoret(message[8:])

if message == 'KENO': return keno()

if message[:8] == 'KONVERTO': return convert(message[9:17], int(message[18:]))

if message[:4]=='AREA': return area(int(message[5:6]), int(message[7:8]))

if message[:8]=='HARMONIK': return harmonik(int(message[9:10]))

return "Komand e panjohur. Shkruani edhe nje here kerkesen."

main()

**KODI PER KLIENT**

from socket import\*

host='localhost'

port=9000

socketClient=socket(AF\_INET,SOCK\_STREAM)

socketClient.connect((host,port))

while 10:

message=input("Shkruani Kërkesën :")

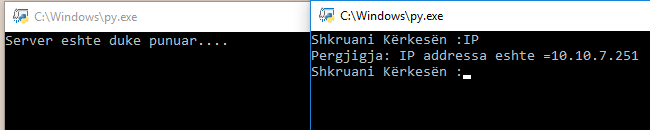
socketClient.send(message.encode('ASCII'))

if\_else\_func=socketClient.recv(128)

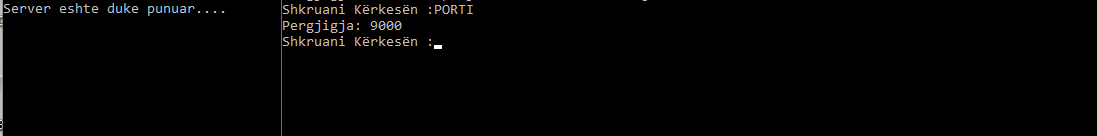
print("Pergjigja: " + if\_else\_func.decode('ASCII'))

**4. Përshrimi i kodit**

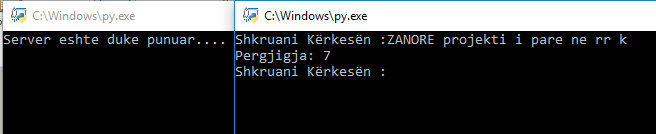
**5.1 Kërkesa IP**



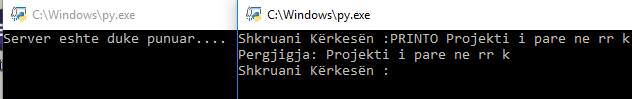
**5.1 Kërkesa PORT**



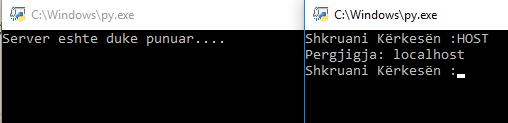
**5.1 Kërkesa ZANORE**



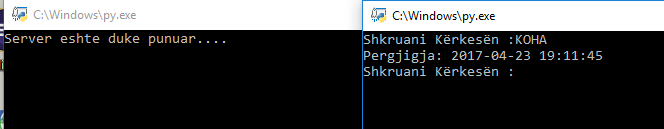
**5.1 Kërkesa PRINTO**



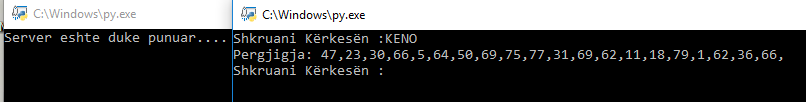
**5.1 Kërkesa HOST**



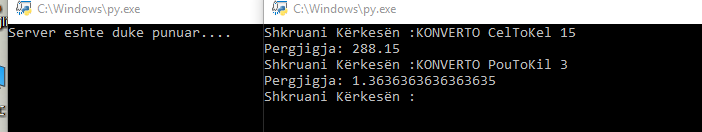
**5.1 Kërkesa TIME**



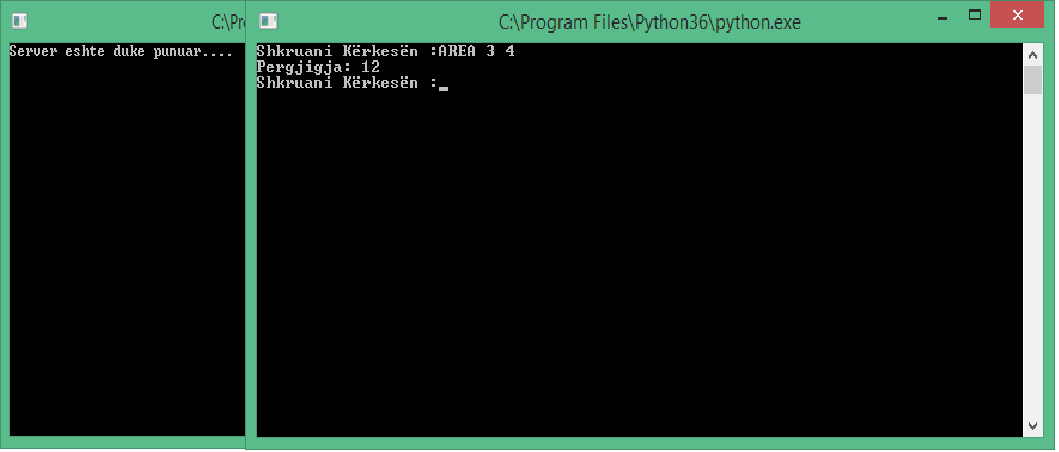
**5.1 Kërkesa KENO**



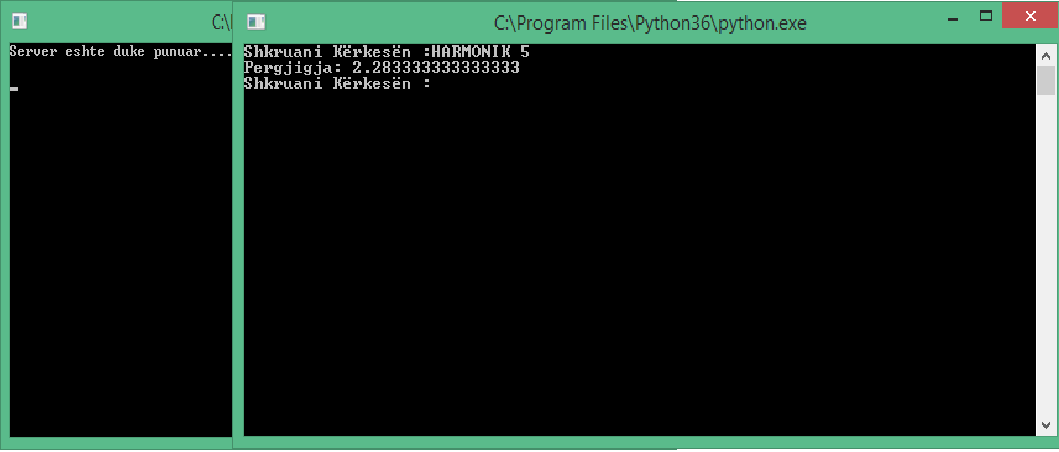
**5.1 Kërkesa KONVERTO**



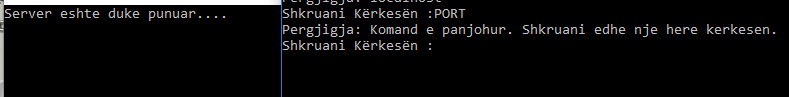
**5.1 Kërkesa AREA**

****

**5.1 Kërkesa HARMONIK**

****

**Rasti kur nuk është asnjëra nga kërkesat.**



.