**RELAZIONE DI PROGETTO**

**PROGRAMMAZIONE DI RETI**

Traccia 2

Alesja Delja – [Alesja.Delja@studio.unibo.it](mailto:Alesja.Delja@studio.unibo.it)

0000969763

Silvia Furegato – [Silvia.furegato@studio.unibo.it](mailto:Silvia.furegato@studio.unibo.it)

0000977475

Tommaso Turci – [Tommaso.Turci@studio.unibo.it](mailto:Tommaso.Turci@studio.unibo.it)

0000??????

**Indice**

**Descrizione generale delle scelte di progetto effettuate……………………........3**

Funzionalità del client……………………………………………………..……....5

Funzionalità del server……………………………………………………..….......8

**Descrizione delle strutture dati utilizzate……………..………………………………..10**

Thread………………………………………………………………………………….…11

Boh…………………………………………………………………………………….14

**Descrizione generale delle scelte di progetto effettuate**

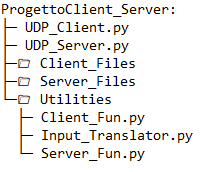
Per poter avviare il programma bisogna avere installato Python e aver clonato il repository <https://github.com/AivAiv/ProgettoClient-Server>. Con il terminale spostarsi fino alla cartella in cui si è clonato il repository e si apre una sceda di terminale per ogni numero di componente che si vuole.

Per eseguire il codice di server/client basta scrivere da terminale, nella cartella in cui sono presenti: python3 *nomefile*.py

Tutti i file che vengono condivisi tra il file e il server vengono salvate in delle cartelle da parte chiamate Client\_Files e Server\_Files (**LE CREIAMO NUOVE QUANDO QUESTE NON ESISTONO OPPURE CI SONO SEMPRE E COMUNQUE**)

Quando si desidera terminare il programma, basta digitare il comando “exit” sul client, che farà chiudere sia il client che il server. In caso ci fossero più client in uso, il primo che verrà chiuso anche il server, lasciando funzionanti i restanti client, i quali dovranno essere chiusi manualmente dall’utente a loro volta. (**Vedere se sta parte la facciamo**)

La cartella è divisa in:







***Funzionalità del client***

Nel file UDP\_Client vengono importato il socket, Input\_Translator e Client\_Fun che servono per implementare al meglio il Client. (Input\_Translator e Client\_Fun si trovano dentro la cartella Utilities.)

**UDP\_Client.py**

Nel file creiamo un nuovo socket impostando come indirizzo IP ‘localhost’ e come porta 10000 che creano così il server\_adress. All’avvio del Client viene subito mostrato il server\_adress e la lista dei comandi che possono essere fatti dal Client stesso; questo input viene poi controllato dal Input\_Translator importato precedentemente.

In base al comando scelto verrà poi eseguita l’operazione scelta che si trovano nel file Client\_Fun. Infine, viene implementato anche il comando exit per chiudere la connessione con il server.

**Input\_Translator.py**

Questo file viene usato per controllare che gli input dati siano validi usando diverse funzioni:

* getFiles ()

prende tutti i nomi dei file presenti nella directory Client\_Files e li metti in una lista.

* checkInput(inString)

divide la stringa, controlla che il comando sia giusto indipendentemente che sia scritto in maiuscolo o minuscolo e **controlla il nome del file.**

* getInput(inString)

ritorna il nome del file e il comando richiesto. **(forse checkInput via)**

**Client\_Fun.py**

Implementa le diverse funzioni che ci servono per il corretto funzionamento del client:

* ListRequest (socket, server\_adress)
* PutRequest (socket, server\_adress, filename)

Entrambi mandano il comando al server e restano in attesa della risposta; PutRequest manda anche il nome del file che si cerca.

***Funzionalità del server***

Nel file UDP\_Client vengono importato il socket e Server\_Fun che servono per implementare al meglio il Server. (Server\_Fun si trova dentro la cartella Utilities.)

**UDP\_Server.py**

Nel file creiamo un nuovo socket impostando come indirizzo IP ‘localhost’ e come porta 10000 che creano così il server\_adress. All’avvio del Server viene subito mostrato il server\_adress.

La funzione client\_handler indirizza a seconda del comando inserito alla funzione che lo implementa dentro Server\_Fun precedentemente importato.

L’exit non viene messo dentro la funzione poiché non è richiamabile dai thread ma solo dal “main”. (**da sistemare**)

**Server\_Fun.py**

Implementa le diverse funzioni che ci servono per il corretto funzionamento del server:

* ListResponse (socket, client\_adress)

Invia la lista al client

* PutResponse (socket, client\_adress)

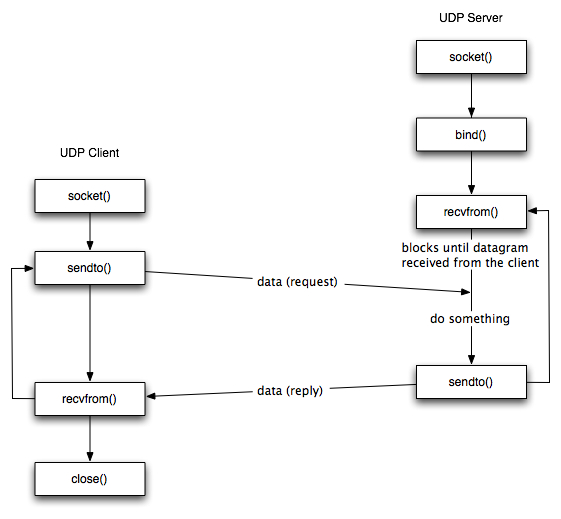
Riceve il file, lo salva nella cartella Server\_Files e ritorna l’esito dell’operazione.

* getFiles ()

controlla se la cartella di destinazione esiste già, se non esiste la crea e poi mette i nomi dei file in una lista

**Descrizione delle strutture dati utilizzate**

L’immagine mostra come il server e il client interagiscono tra loro:



SPIEGAZIONE DEI THREAD