Aplicativo ChatRoom com Servidor e Cliente

Lucas S Schiavini
lucaschiavini@hotmail.com

***Resumo*— Criar um software que liga usuários a um servidor. Esse software deve permitir criação de salas, envio de mensagens e arquivos entre usuários numa mesma sala.**

I. Objetivos

O objetivo do trabalho é desenvolver uma aplicação com os conhecimentos da matéria de forma a ligar clientes em um servidor de forma a trocar mensagens e arquivos entre os clientes.

II. Introdução Teórica

Aplicativos de comunicação existem desde o início da internet. Projetos que permitem troca de mensagens utilizando a rede e um protocolo de comunicação foram cada vez mais adotados pelo público consumidor.

Aplicativos como Skype, WhatsApp, Telegram se encaixam dentro desse conceito.

Em sua base aplicativos de comunicação seguem o padrão de funcionalidades mínimas. Estas sendo: Criação de Usuários, Criação de Salas, Remoção de Salas, trocas de mensagens entre usuários online dentro de uma mesma sala.

A ideia desse projeto[1] é implementar uma versão primitiva dos antigos aplicativos de comunicação estilo "Bate Papo em Sala"que são base da maioria dos aplicativos comerciais atuais.

A. Tecnologias utilizadas

De forma a manter uma comunicação mais próxima do tempo real, sem perda de informações, foram utilizados conceitos de Threads e comunicação por Socket.

1) logging: [3] Uma biblioteca do python que facilita a criação e armazenamento de logs do sistema e do programa, separando em DEBUG, INFO e ERRO.

2) os.path: [2] Uma biblioteca padrão Linux que verifica caminhos de arquivos e diretórios.

3) Socket: [5]

Comunicação por Sockets é o padrão da comunicação de aplicativos, que permite que um nó na rede se comunique com outros.

Com a utilização de Sockets é possível a implementação de TCP na comunicação entre um servidor e um cliente. A forma de utilizar esse protocolo foi com a opção SOCK_STREAM no objeto socket do Python.

```
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Fig. 1. Python Socket

4) Threads: [4]

Threads permitem que o programa emule um paralelismo de forma a não ficar preso em um trecho do código e perder informações que são necessárias para outra seção.

Dessa forma é possível, com apenas um programa Servidor, existirem vários canais, cada um comunicando com um servidor, sem vazamento de dados entre um canal e outro.

III. Descrição Experimental

O aplicativo foi desenvolvido em Python 3.6. O escopo do projeto engloba a implementação de um servidor e de um cliente de forma a fazer a comunicação.

O funcionamento, de forma abrangente, foi descrito de forma de máquina de estados finitos nas figuras 2 e 3.

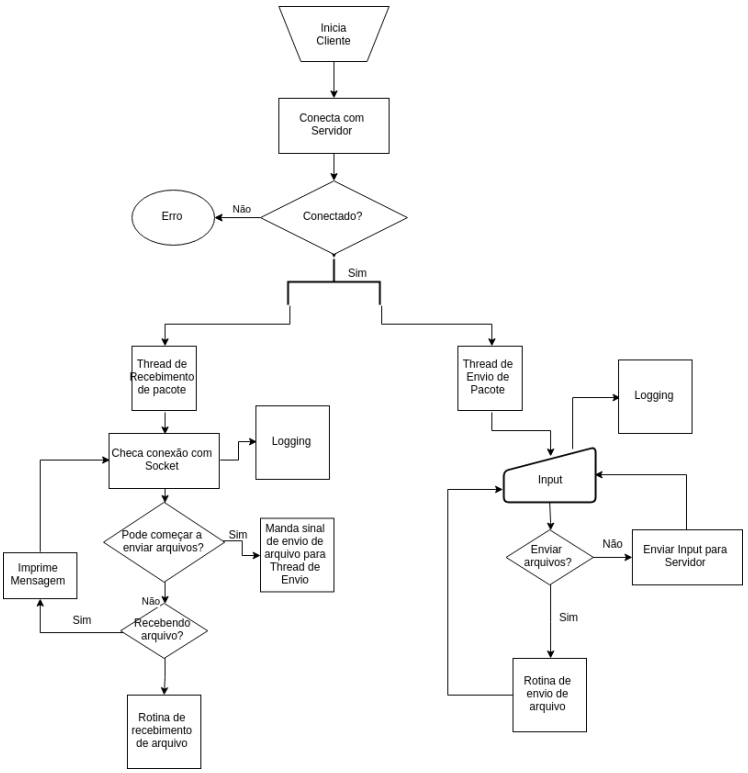

Fig. 2. Máquina de Estados Finitos: Cliente

É possível verificar que o cliente utiliza duas Threads, de forma a conseguir emular paralelismo ao enviar e receber mensagens do servidor. Além disso há armazenamento de dados num arquivo de log tanto ao receber mensagens quanto no envio.

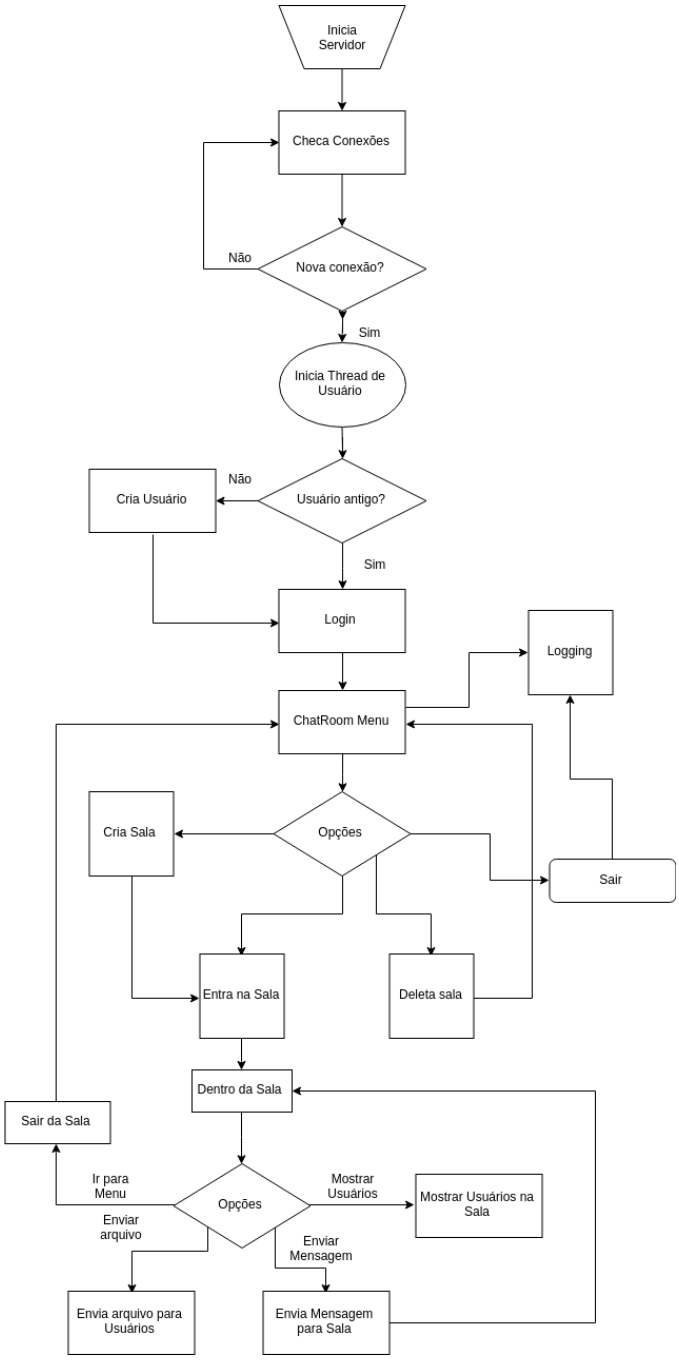

Fig. 3. Máquina de Estados Finitos: Servidor

No lado do servidor, há uma checagem constante por novas conexões, e para cada nova conexão, uma thread é iniciada para lidar com a criação de usuário, login e operação do aplicativo daquela conexão.

A. *Login*

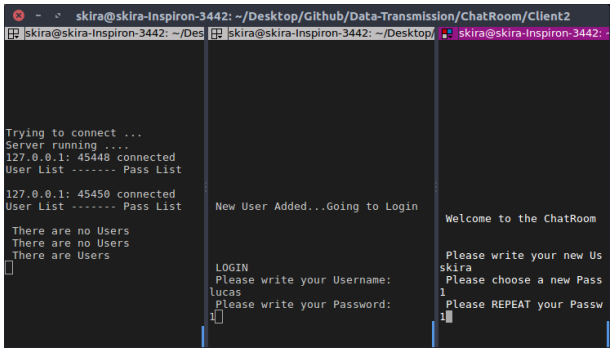

Fig. 4. Estado: Criando usuário

É possível ver a criação de um usuário na figura 4. É necessário confirmar a senha de forma a finalizar a criação. Caso contrário, o usuário não será criado e o programa volta ao estado de Criar Usuário.

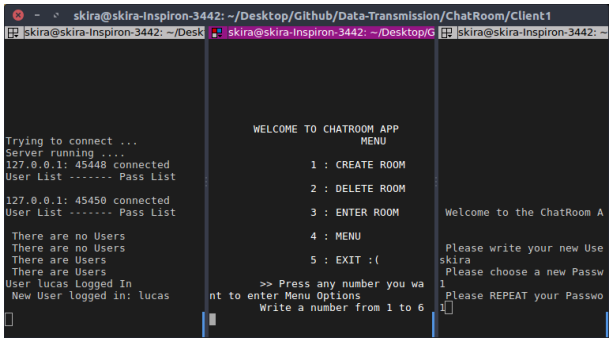

Fig. 5. Estado: Login

Na figura 5 é possível observar no servidor(Terminal mais à esquerda) que foi criado um usuário, e que esse usuário entrou no programa.

Durante o login o programa testa se a senha introduzida é a mesma cadastrada anteriormente.

B. *Menu*

De forma a simplificar a interface, após o Login, o servidor envia as opções de menu para o cliente a cada vez que o usuário estiver fora de uma sala.

As opções principais do Menu são Criar Sala, Deletar Sala, Entrar na Sala e Sair do programa.

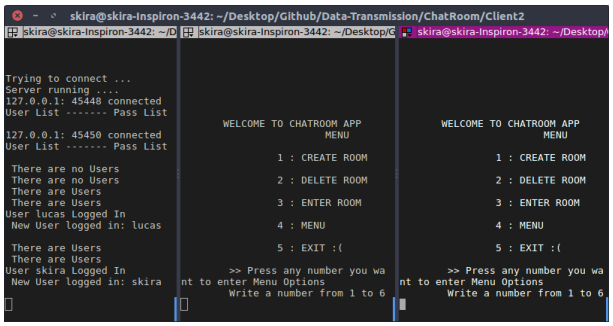

Fig. 6. Estado: Menu Principal

Na figura 6 é possível ver as opções citadas anteriormente. Assim como uma adição de outro usuário(skira).

C. Criar Sala

A primeira opção do Menu é Criar Salas. Nessa opção podemos criar salas com ou sem senha. Salas com senha só podem ser acessadas por usuários que possuam essa senha.

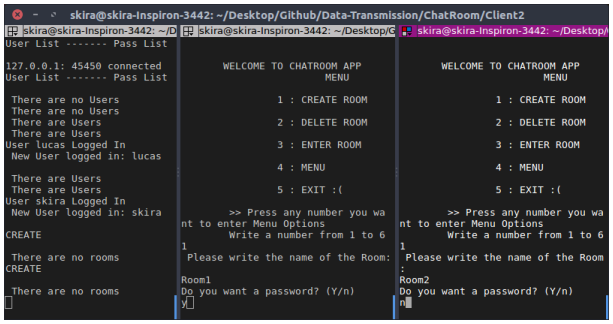

Fig. 7. Estado: Criando Sala

Na figura 7 temos o exemplo do início da criação de uma sala, à esquerda o Servidor mostra que ainda não há salas, mas que está executando a função de criação de salas. No meio temos a criação de uma sala com senha, à direita a criação simples.

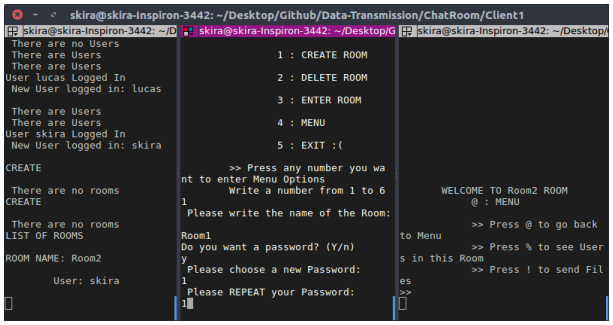

Fig. 8. Estado: Criando Sala Com Senha

Na figura 8 temos a criação de uma sala com senha, e à direita o usuário já entrou na sala simples.

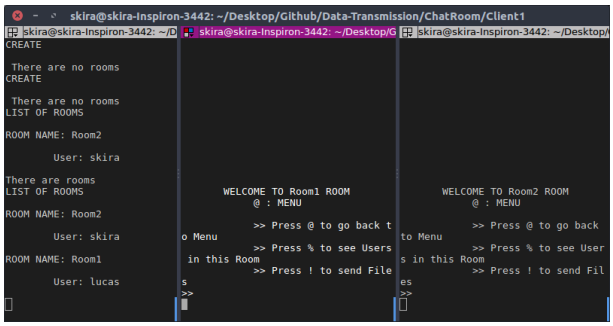

Fig. 9. Estado: Dentro das Salas

Finalmente, na figura 9 vemos que o servidor criou as salas e já armazenou os usuários dentro de cada sala respectiva.

Como cada usuário está em uma sala diferente, mensagens que um envia não será vista pelo outro. Podemos ver isso na figura 10 a seguir.

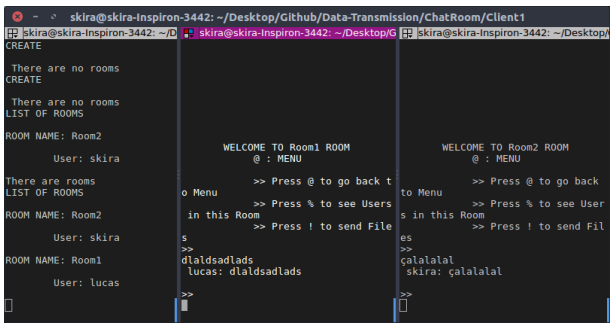

Fig. 10. Estado: Mensagens em salas diferentes

D. Entrar na Sala

De forma a dois usuários se comunicarem, é necessário que ambos estejam na mesma sala. Assim, temos a operação 3 do Menu Principal, que é Entrar em Sala. De forma a voltar ao menu principal, basta escrever '@' no terminal.

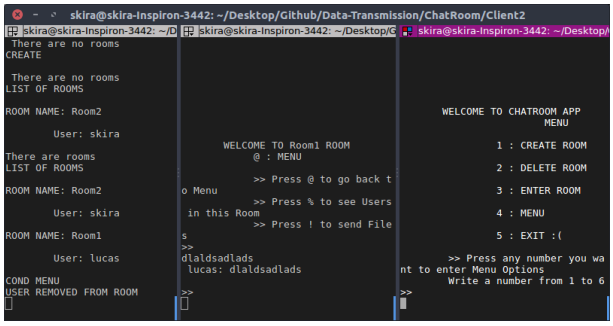

Fig. 11. Estado: Saindo da Sala

Na figura 11 vemos que o servidor removeu o usuário da sala Room2.

De forma a entrar na sala do primeiro usuário(lucas), deve-se escrever 3.

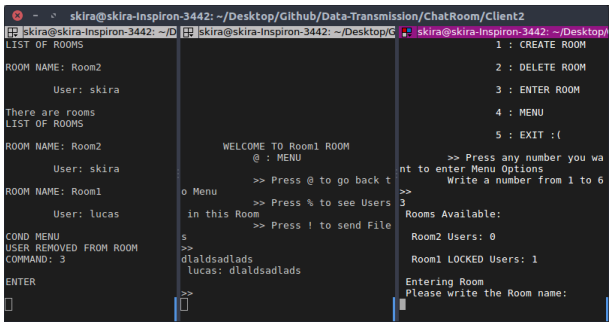

Fig. 12. Estado: Entrando na Sala

É possível ver na figura 12 que há 2 salas, uma com senha(LOCKED) e outra sem. É possível ver ainda o número de usuários em cada sala.

Nessa etapa o usuário pode entrar em quaisquer salas que existam, caso digite uma sala que não exista o programa pedirá que o usuário execute outra ação.

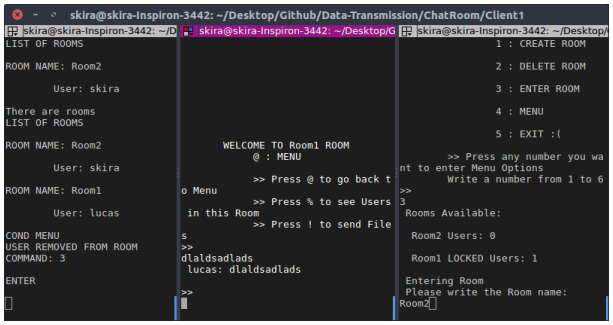

Fig. 13. Estado: Entrando na Sala sem Senha

Ao tentar entrar na sala sem senha Room2 na figura 13 basta escrever o nome da sala.

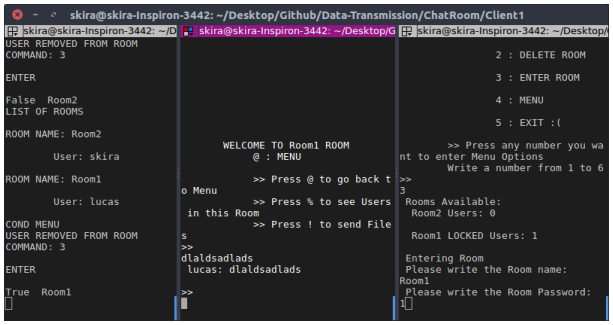

Fig. 14. Estado: Entrando na Sala com Senha

Já para entrar na sala Room1, será requisitada a senha, como pode ser visto na figura 14.

E. Dentro da Sala

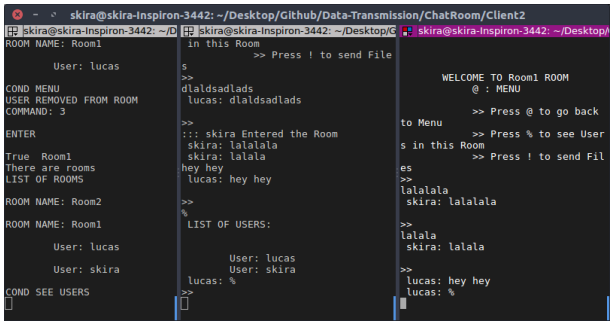

Fig. 15. Estado: Dentro da Sala

É possível perceber 3 informações importantes na figura 15. No lado do servidor(Painel mais à esquerda), é possível ver as Salas e os usuários em cada. No painel do meio, é possível ver que o outro usuário(skira) entrou na sala. Assim como as mensagens de skira são visíveis, as de lucas também são para skira. Finalmente, é possível ver que utilizando o comando '%' a lista de usuários naquela sala aparece para lucas.

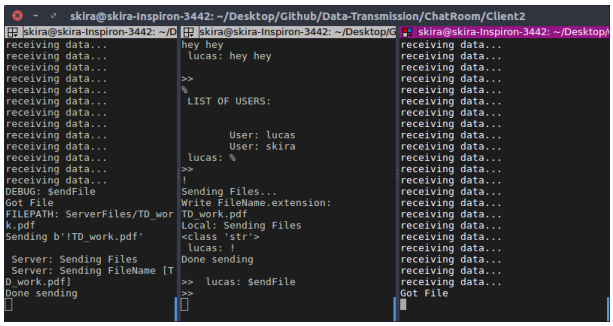

Fig. 16. Estado: Enviando Arquivos

Uma última ação que pode ser feita na sala é o envio de arquivos. Pode ser visto na figura 16 que o servidor recebe o arquivo do primeiro cliente(lucas) para depois reenviar para o segundo(skira).

Caso o usuário não tenha a pasta LocalFiles onde o arquivo pode ser salvo, o programa cria essa pasta automaticamente antes de receber o arquivo.

F. Deletar Sala

De forma a deletar uma sala, é necessário voltar ao menu principal e selecionar a opção 2.

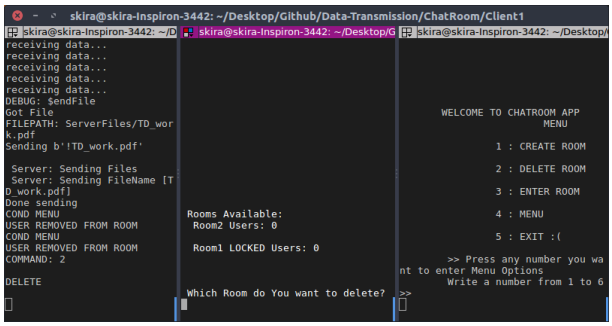

Fig. 17. Estado: Deletando Salas

Na figura 17 vemos as duas salas disponíveis para deleção. Room1 com senha e Room2 sem senha.

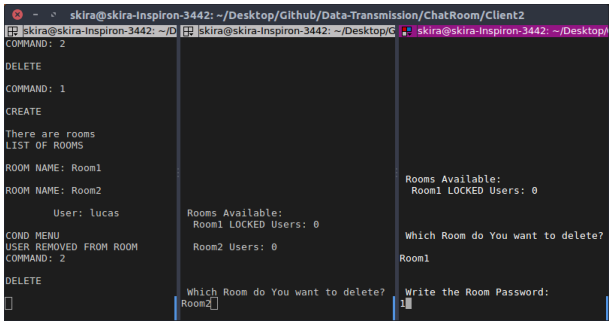

Fig. 18. Estado: Deletando Salas com e sem Senha

É possível verificar na figura 18 que para deletar uma sala com senha é requisitada a senha, e para salas sem senha, basta escrever o nome da sala que deseja-se a deleção. Finalmente temos na figura 19 a deleção das duas salas vistas no terminal do servidor(à esquerda).

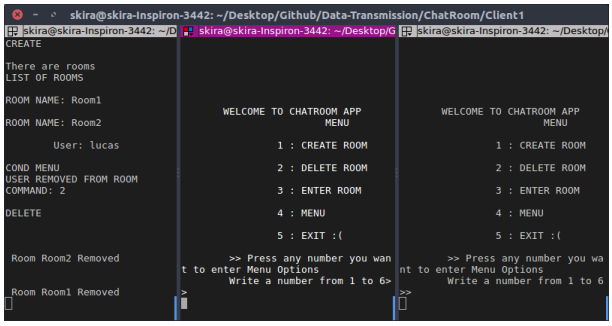

Fig. 19. Estado: Salas deletadas

IV. Resultados

Os resultados foram compilados em figuras de cada estado do projeto, assim como os diagrama de máquina de estados. Essas informações podem ser obtidas na pasta /Documentation do diretório[6] do projeto. Informações de como executar o código estão no ReadMe na pasta root do projeto.

A conexão servidor cliente funcionou em ambos os códigos, e foi separada em 3 pastas: Cliente1, Cliente2, e Servidor.

A. Problemas Encontrados

Alguns dos problemas encontrados se deram na sincronização de códigos que necessitavam operar em paralelo, como de enviar e receber mensagens do servidor. A forma mais simples utilizada para resolver tal problema foi o uso da biblioteca threading que paraleliza em nível de software o programa.

Além disso, erros de conexão criavam exceções e blocos de Try Exception no Python foram necessários para solucionar esses erros.

Recebimento de mensagens em outros formatos. Foram utilizadas as funções decode() e encode() do Python para manter mensagens enviadas e recebidas no mesmo formato "utf-8".

Grande parte dos problemas se deu no recebimento e envio de arquivos do servidor para o cliente, de forma que foi necessária a implementação de um protocolo para indicar que o servidor ou o cliente estavam prontos para receber o nome do arquivo, e só então enviar um sinal pra começar a transferência de arquivos. Antes foi implementada uma versão que extraia o nome do arquivo do próprio binário do arquivo.

V. Discussão e Conclusões

Com essa atividade, foi possível entender melhor o funcionamento de uma conexão simples de um servidor e cliente, assim como a importância de paralelização em aplicativos comuns, como os de comunicação que são amplamente utilizados.

Foi possível entender ainda, o processo de produção de softwares na camada de aplicação, e como esta facilita os processos de comunicação.

Além disso criou-se uma apreciação sobre protocolos na camada de transporte como TCP, visto que o programa foi criado de forma a só enviar novos dados quando a conexão retornasse um sinal de que recebeu os pacotes anteriores.

References

[1] Descrição e requisitos do Trabalho, `Arquivo_TD_201802_Trabalho_Final.pdf`, 2018.
[2] Documentação do módulo os.path, `<https://docs.python.org/2/library/os.path.html>`, 2018.
[3] Documentação do módulo logging, `<https://docs.python.org/2/library/logging.html>`, 2018.
[4] Documentação do módulo threading, `<https://docs.python.org/2/library/threading.html>`, 2018.
[5] Documentação do módulo socket, `<https://docs.python.org/2/library/socket.html>`, 2018.
[6] Repositório do Git, `<https://github.com/lschiavini/Data-Transmission>`, 2019.