INSTITUTO DO EMPREGO

E FORMAÇÃO PROFISSIONAL SETÚBAL

ESTRUTURA DE DADOS ESTÁTICA, COMPOSTA E DINÂMICA

TEGRSI13

PROJETO TFTPy: CLIENTE E SERVIDOR TFTP

Tamires Claro Juliandro Lessa

Fevereiro/2025

Sumário

Introdução e Objetivos

Este projeto tem como objetivo a implementação de um **cliente TFTP (Trivial File Transfer Protocol)** utilizando **Python 3** e a biblioteca **socket** para comunicação em rede.

O **TFTP** é um protocolo simples para transferência de arquivos via **UDP**, muito utilizado para tarefas como **boot remoto de sistemas operacionais**. Diferente do **FTP**, ele não possui autenticação e trabalha com pacotes de **512 bytes**, seguindo um modelo de comunicação chamado **stop-and-wait**, onde cada bloco de dados precisa ser confirmado antes de enviar o próximo.

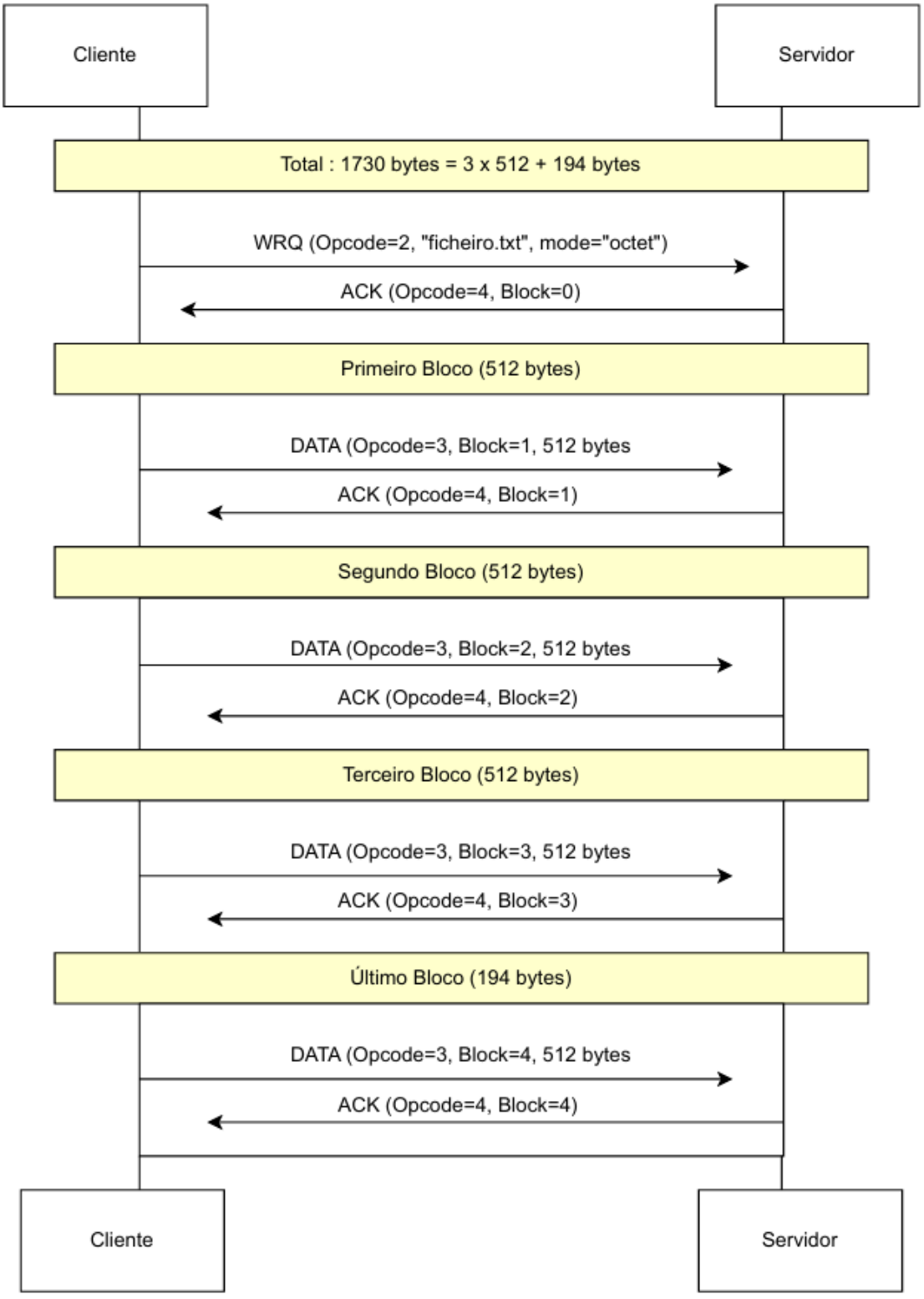
Nosso cliente foi desenvolvido para funcionar de duas formas:

- **Modo interativo** – Permite ao usuário enviar e receber arquivos manualmente, através de comandos dentro do próprio cliente.
- **Modo não interativo** – Permite executar operações diretamente na linha de comando, sem precisar acessar um menu interativo.

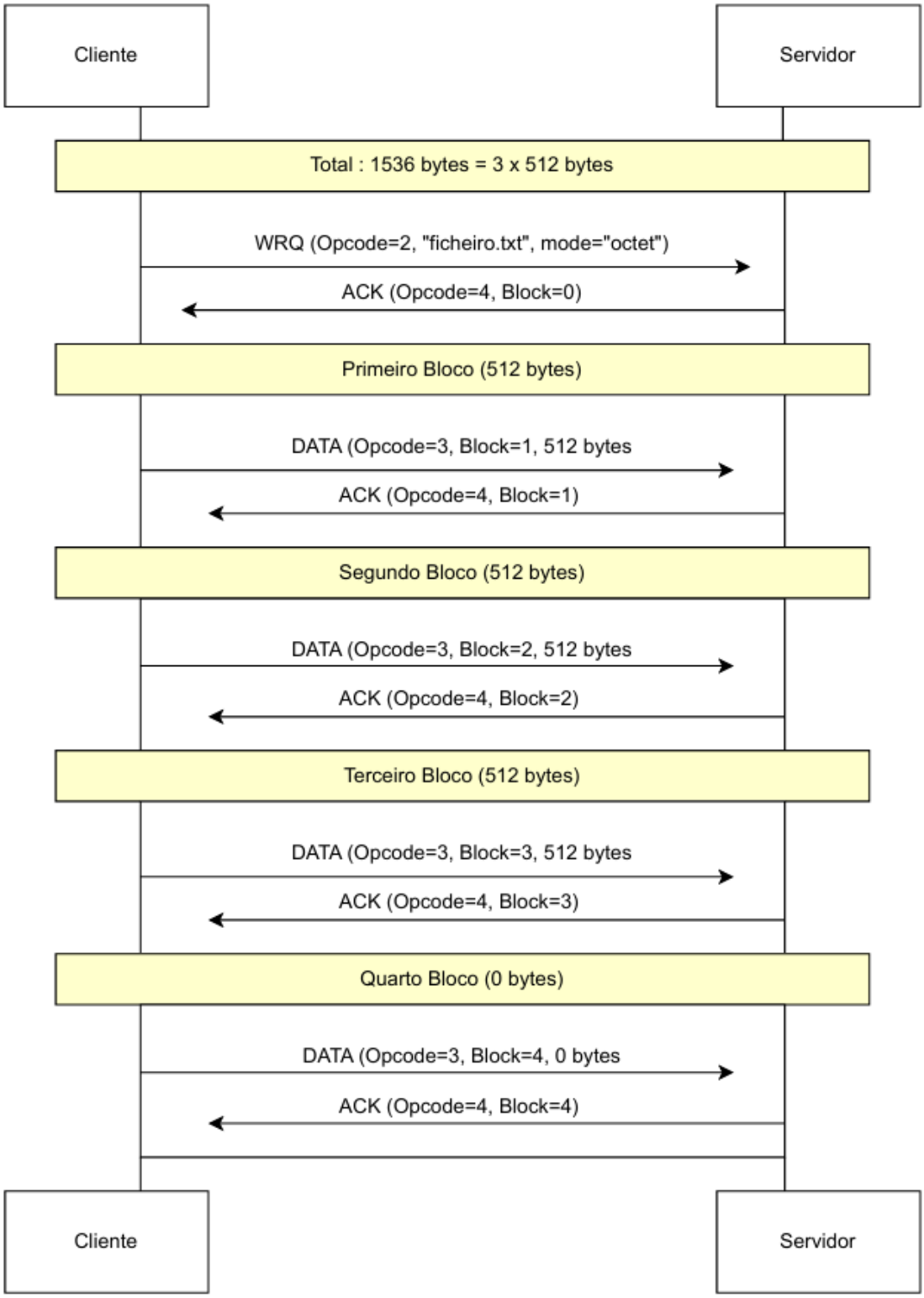
Com essa abordagem, o projeto não apenas implementa o protocolo **TFTP**, mas também cria uma interface prática e fácil de usar, tornando a experiência do usuário mais intuitiva.

Análise

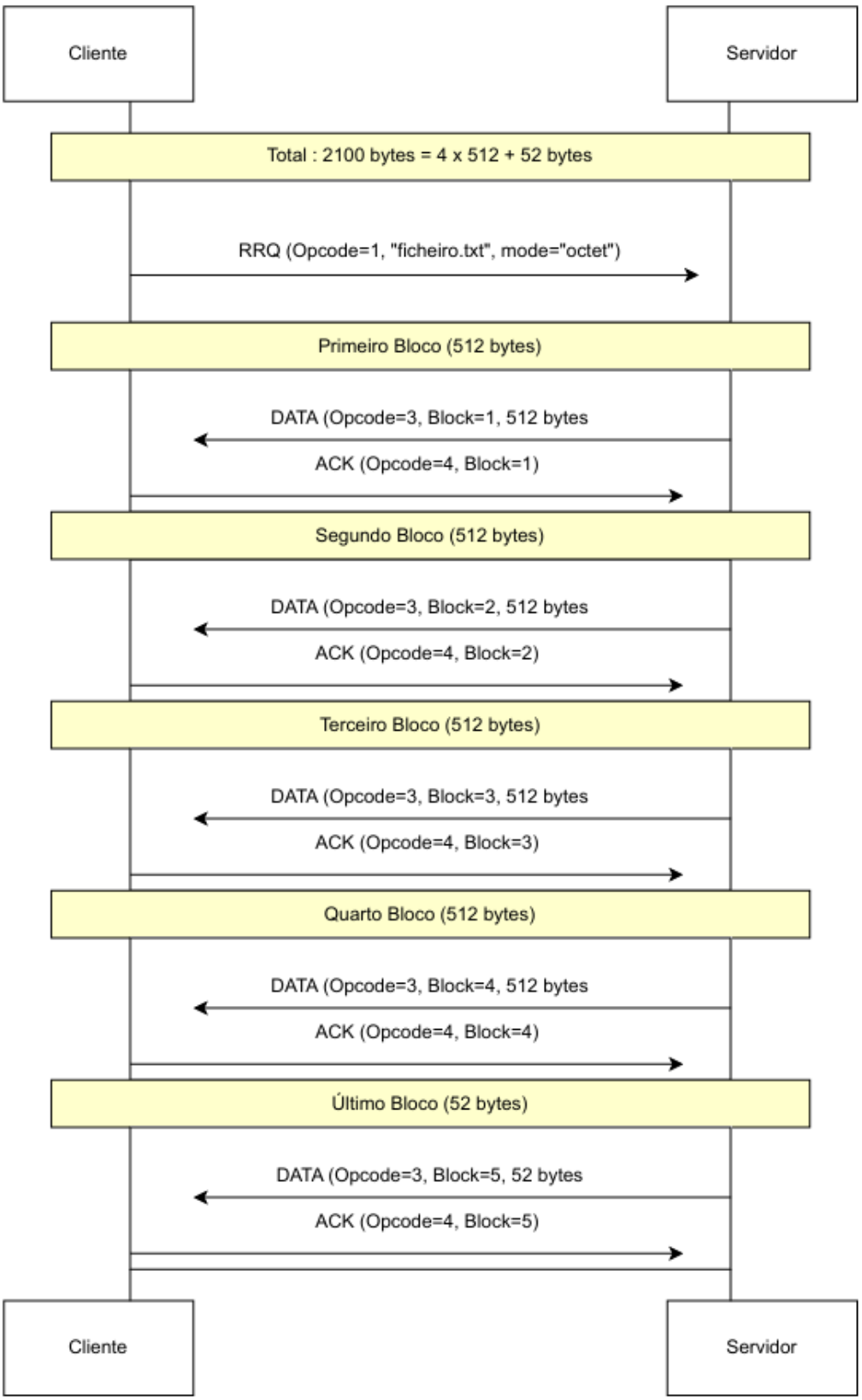
Envio de um ficheiro com 1730 bytes:



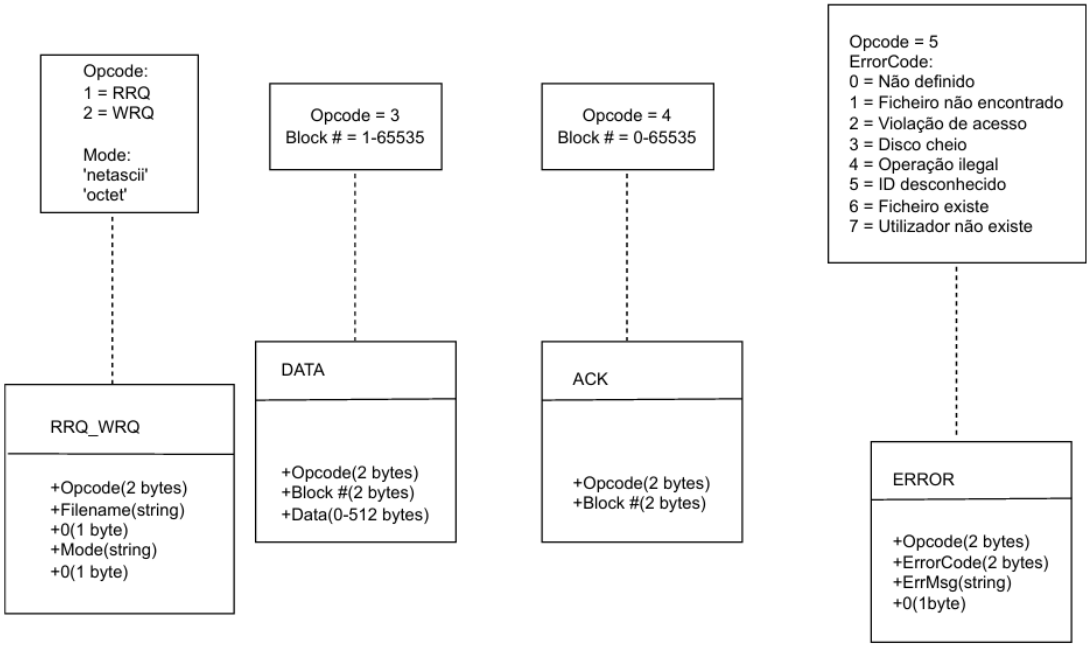
Envio de ficheiro com 1536 bytes:



Envio de um ficheiro com 2100 bytes:

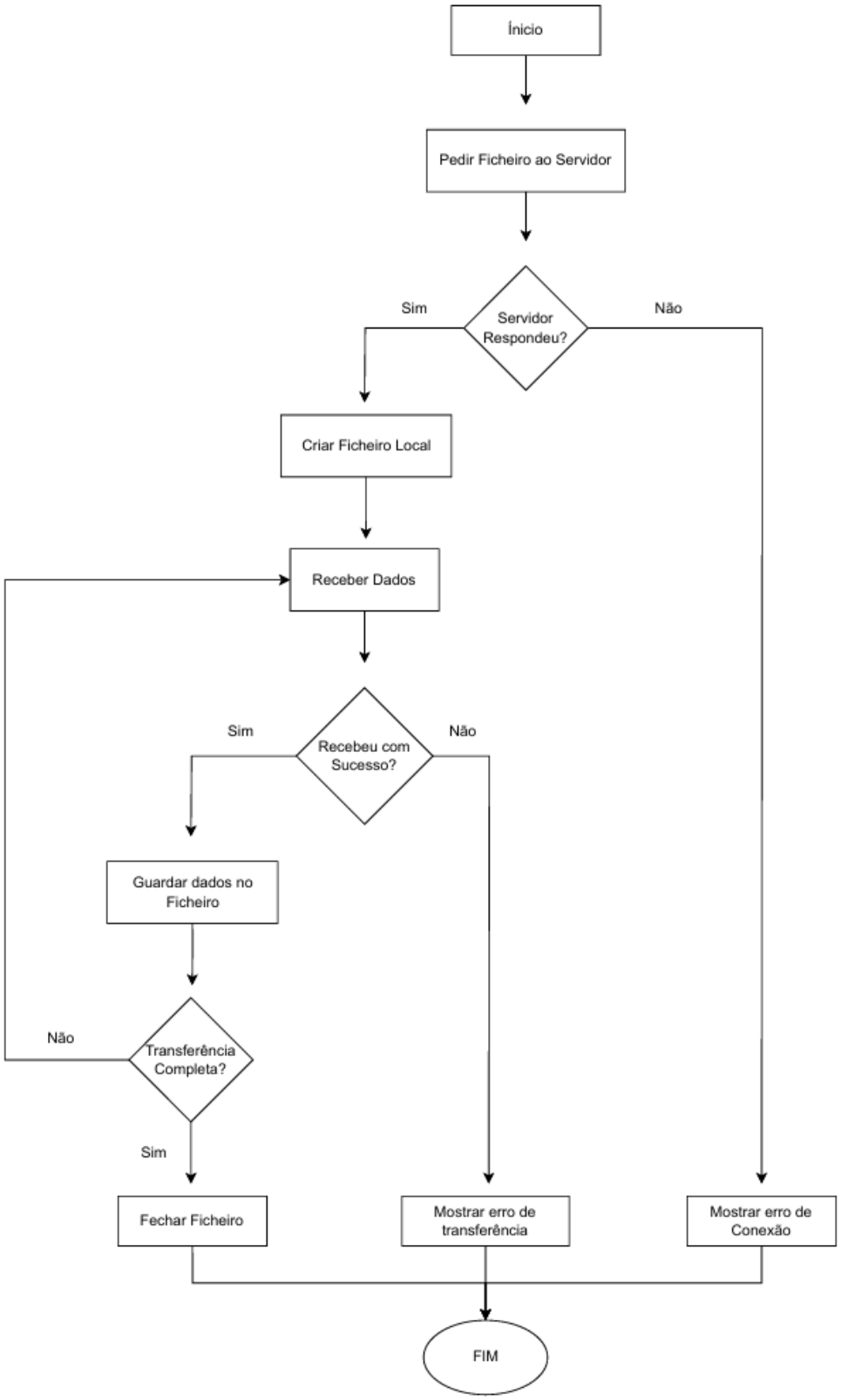


Formato dos pacotes:

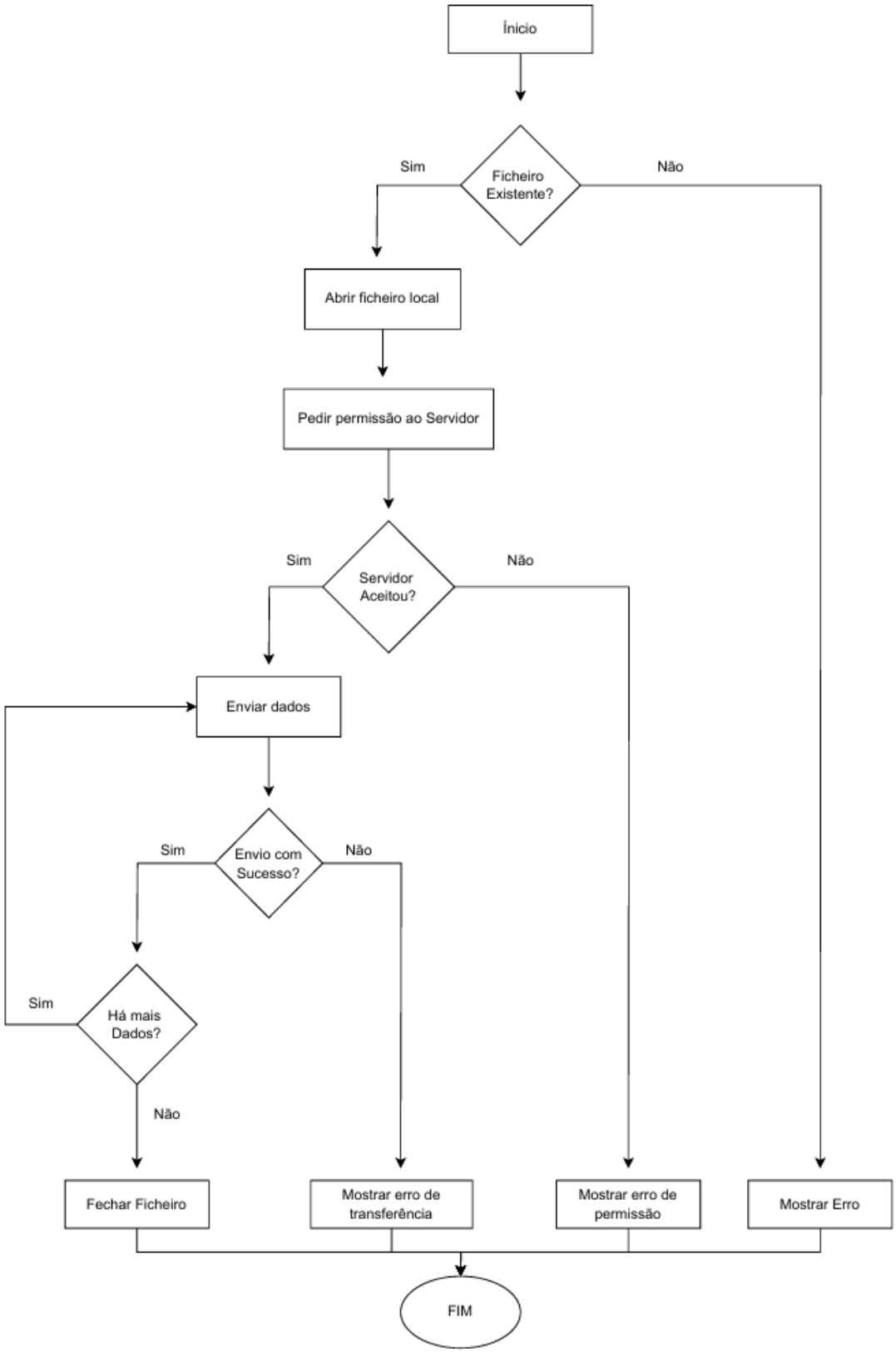


Desenho e Estrutura

Fluxograma RRQ:



Fluxograma WRQ:



Implementação

O projeto **TFTPY13** foi desenvolvido para implementar um cliente TFTP, permitindo a transferência de arquivos entre um cliente e um servidor de forma simples e eficiente. O código-fonte do projeto está armazenado na pasta **Projeto2**, que contém os seguintes componentes principais:

- **`src`**: Diretório onde está localizado o código do cliente TFTP e seus módulos auxiliares.
- **`docs`**: Pasta que armazena a documentação do projeto.
- **`venv`**: Ambiente virtual do Python, garantindo que todas as dependências sejam instaladas de maneira isolada.
- **`.vscode`**: Arquivos de configuração do Visual Studio Code para facilitar o desenvolvimento.

Cliente TFTP (`client.py`)

O arquivo **client.py** contém a implementação do cliente TFTP, permitindo que os usuários interajam com o servidor de duas formas:

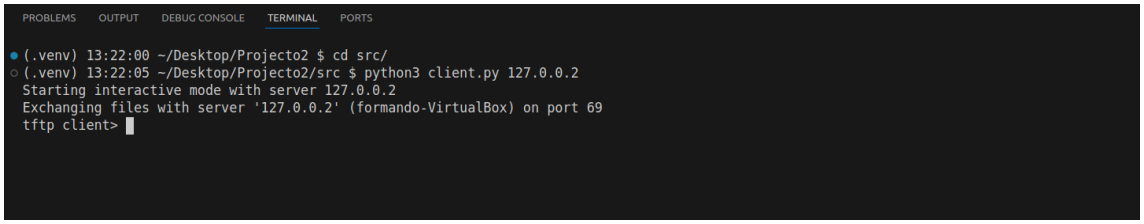
1. **Modo Interativo**: O cliente inicia um terminal próprio onde o usuário pode inserir comandos manualmente, como `get` para baixar arquivos e `put` para enviá-los.

2. **Modo Não Interativo**: O cliente recebe comandos diretamente da linha de comando, executando operações automaticamente sem a necessidade de interação contínua do usuário.

Durante a execução, o cliente lida com diferentes situações, incluindo erros de conexão, arquivos inexistentes no servidor e problemas de rede.

1-Modo Interativo

No modo interativo, o cliente inicia um terminal próprio, exibindo uma interface como esta:



A partir dessa interface, o usuário pode utilizar os seguintes comandos:

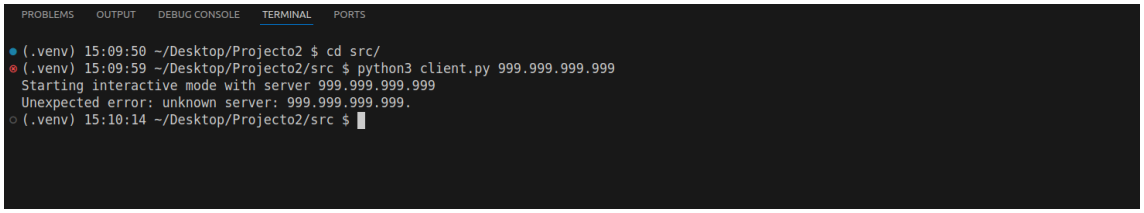
2-Modo Não Interativo

O modo não interativo permite executar comandos diretamente no terminal:

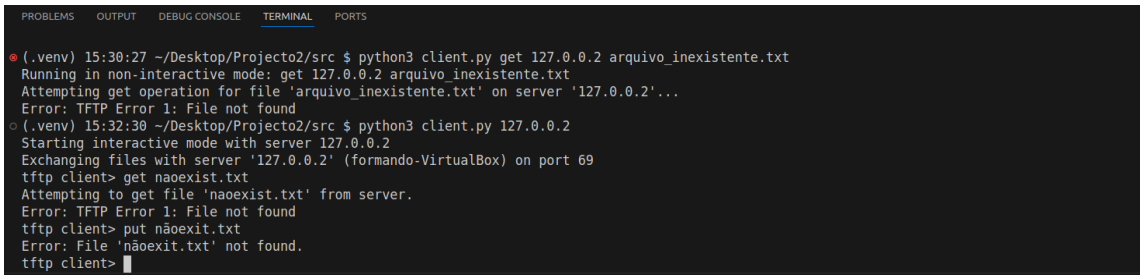
Tratamento de Erros

Sabemos que em redes, falhas podem acontecer. Por isso, tanto o nosso cliente interativo como o não interativo lida com erros como:

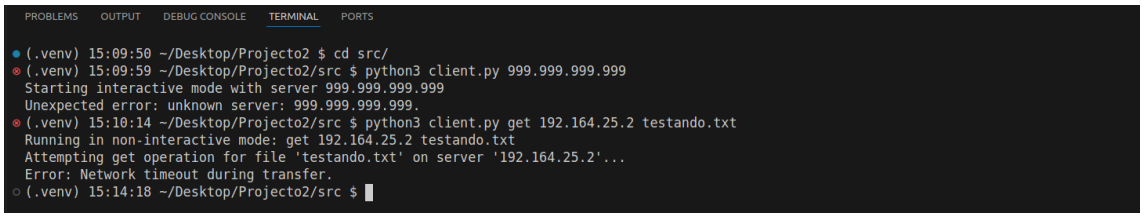
- **Servidor não encontrado:**



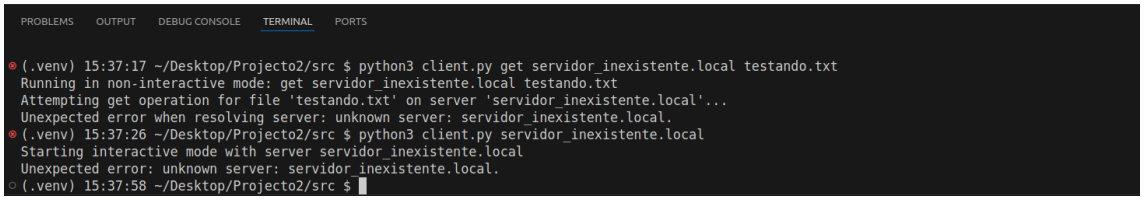
- **Arquivo não encontrado no servidor:**



- **Problema de Rede ou Timeout:**



- **Erro inesperado (exemplo, permissão negada):**



Isso garante uma experiência mais amigável para o usuário, evitando falhas silenciosas.

Módulo `tftp.py`

O arquivo **tftp.py** contém funções auxiliares que lidam diretamente com a comunicação entre o cliente e o servidor. Ele implementa a lógica de envio e recebimento de pacotes TFTP, seguindo o protocolo adequado para garantir que os arquivos sejam transferidos corretamente.

Entre as principais funções do módulo, destacam-se:

- **`get_file(server_addr, filename)`**: Responsável por solicitar o download de um arquivo do servidor.
- **`put_file(server_addr, filename)`**: Envia um arquivo do cliente para o servidor.
- **`get_host_info(server)`**: Obtém o endereço IP do servidor a partir do nome ou IP informado.

Esse módulo garante que o cliente consiga se comunicar corretamente com o servidor TFTP, tratando erros de protocolo e problemas de rede quando necessário.

Ambiente Virtual (venv)

O uso do **Ambiente Virtual Python (venv)** é essencial para garantir que todas as bibliotecas do projeto **sejam instaladas de forma isolada**, evitando conflitos entre pacotes e versões diferentes no sistema. Isso é particularmente útil em projetos colaborativos, onde diferentes desenvolvedores podem ter versões diferentes de dependências instaladas.

Para criar um ambiente virtual, utilizamos o seguinte comando:

```
python3 -m venv .venv
```

Esse comando cria um diretório chamado **.venv/** dentro do projeto, que contém uma cópia isolada do interpretador **Python** e das ferramentas de gerenciamento de pacotes (**pip**).

Para ativar o venv No Linux/macOS:

```
source .venv/bin/activate
```

No Windows (cmd):

```
.venv\Scripts\activate
```

Após a ativação, o terminal mostrará:

```
(.venv) usuario@maquina:~/Desktop/Projecto2/src$
```

O prefixo (.venv) indica que o ambiente virtual está ativo e pronto para instalar dependências.

Dentro do ambiente virtual, instalamos as bibliotecas do projeto com:

```
pip install -r requirements.txt
```

O arquivo requirements.txt deve conter todas as dependências do projeto.

Isso garante que todos os desenvolvedores tenham **as mesmas versões de pacotes** e que o código funcione corretamente em qualquer máquina.

Para sair do ambiente virtual:

```
deactivate
```

Se for necessário remover e recriar o ambiente virtual:

```
rm -rf .venv # Linux/macOS

rd /s /q .venv # Windows (cmd)

python3 -m venv .venv

source .venv/bin/activate # Reativar

pip install -r requirements.txt
```

Conclusão

A implementação do **cliente TFTP** neste projeto permitiu a exploração prática de conceitos fundamentais de **transferência de arquivos via rede**, utilizando o protocolo **TFTP (Trivial File Transfer Protocol)** sobre **UDP**. A estrutura do código foi organizada para suportar tanto **modo interativo** quanto **modo não interativo**, permitindo que o usuário escolha a forma mais conveniente para realizar suas operações de envio e recebimento de arquivos.

Além da implementação funcional, foi dada atenção especial ao **tratamento de erros**, garantindo que falhas comuns, como **tempo de resposta excedido, arquivos inexistentes no servidor e problemas de conexão**, fossem devidamente tratadas e exibidas ao usuário de maneira clara. Os testes executados comprovaram a robustez do cliente, validando sua capacidade de lidar com diferentes situações em um ambiente de rede.

Para garantir a **portabilidade e a integridade do ambiente de desenvolvimento**, foi utilizado um **ambiente virtual (venv)**, permitindo que todas as dependências fossem gerenciadas isoladamente. Isso assegura que o cliente TFTP possa ser executado em diferentes máquinas sem riscos de incompatibilidade.

Por fim, a documentação do projeto, incluindo **prints das execuções**, demonstra o funcionamento do cliente e como ele interage com o servidor TFTP. A estrutura organizada e modular do código facilita futuras expansões e aprimoramentos, tornando este projeto uma base sólida para estudos e implementações mais avançadas na área de redes e sistemas distribuídos.