Laboratório 4
Aplicação de “bate-papo” distribuído

Aluno: Guilherme Avelino do Nascimento
Email: guiavenas@gmail.com
DRE: 117078497

O objetivo deste documento é apresentar o projeto da aplicação de bate papo distribuído, detalhando o formato das mensagens de entrada do usuário, as arquiteturas de sistema e de software, e o protocolo a ser utilizado na troca de mensagem entre os processos da aplicação.
Em relação a primeira versão do documento, foram acrescentadas as alterações realizadas nas seções de 1 a 4, e foi incluída a seção 5, apresentando recomendações para a execução do projeto.

Seções:

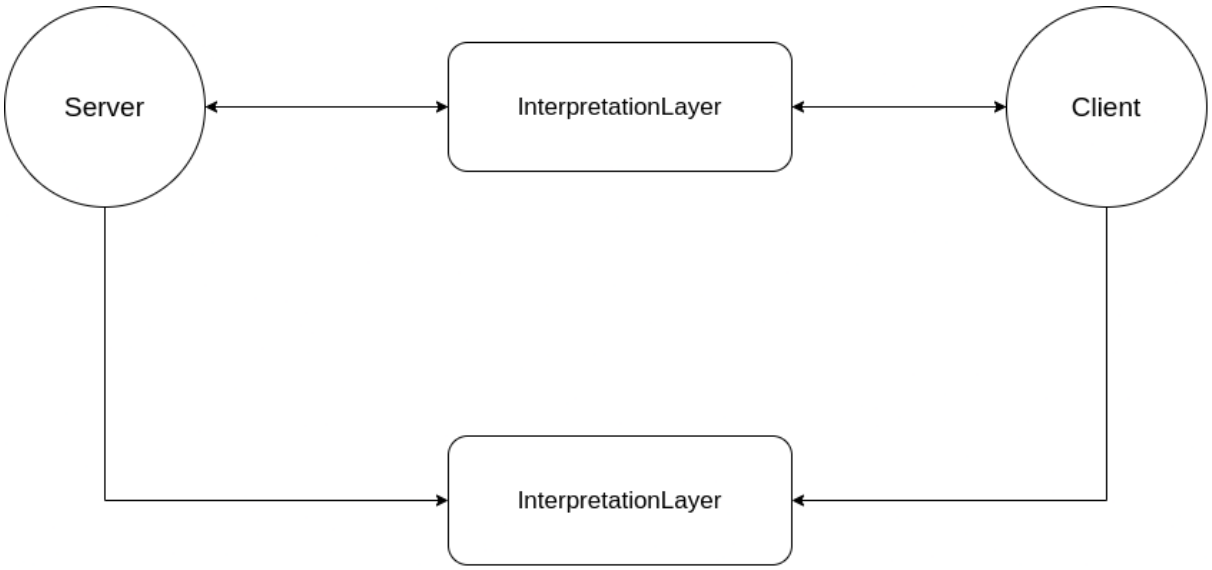
1 - Interface com o usuário

O usuário poderá interagir com a aplicação por meio dos seguintes comandos:

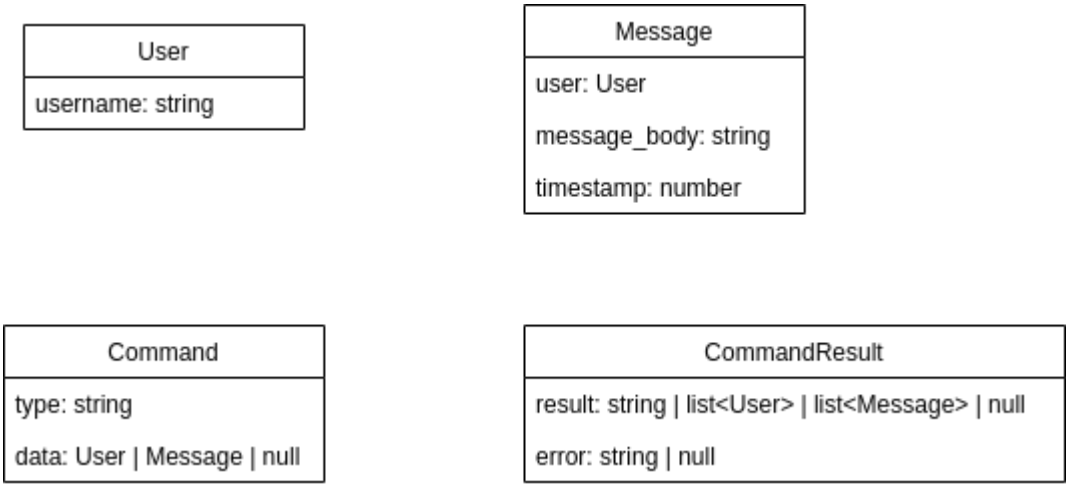
- **HELLO {username}**: Indica que o usuário está disponível para troca de mensagens.
- **BYE**: Torna o usuário indisponível para troca de mensagens.
- **PEERS**: Lista os demais usuários que se encontram disponíveis para troca de mensagens.
- **MESSAGE {username} {message_body}**: Envia uma mensagem para um usuário.
- **LIST**: visualiza as mensagens recebidas.
- **QUIT**: encerra a aplicação(cliente ou servidor).

2 - Arquitetura de software

Diferentemente do proposto no projeto da aplicação, a arquitetura de software está melhor classificada como ‘em objetos’; Os objetos Client e Server utilizam os serviços providos pelos objetos InterpretationLayer(responsável pela interpretação dos comandos digitados via linha de comando no cliente/servidor) e CommunicationLayer(responsável pela comunicação entre os processos). A figura a seguir representa melhor essa organização:



As principais interfaces serão as seguintes: Message, User, Command(requisição que irá trafegar entre processos) e CommandResult(resultado das requisições). As propriedades de cada uma estão descritas no seguinte diagrama:



Além disso, foram adicionados enums para indicar o tipo do comando(com a lista de comandos apresentada na seção 1), e para indicar o status do usuário(ONLINE, INACTIVE, E OFFLINE).

3 - Arquitetura de sistema

A aplicação será implementada na arquitetura cliente-servidor. O tanto o processo cliente quanto o servidor utilizam os objetos CommunicationLayer e InterpretationLayer. O processamento de cada comando é implementado no cliente e servidor diferentemente.

A interação entre os processos seguirá o modelo cliente-servidor padrão: os clientes irão requisitar ao servidor o envio de mensagens e os demais comandos descritos na seção 1, e

o servidor realizará o processamento de localização dos destinatários e envio das mensagens de fato.

O projeto poderá ser utilizado instanciando os processos do cliente e do servidor na máquina local, ou em diferentes máquinas(desde que o IP privado dessas máquinas esteja liberado para comunicação).

4 - Protocolo e formato das mensagens entre processos

Os processos deverão trocar mensagens por intermédio do protocolo TCP na camada de transporte.

Ocorreu uma mudança em relação o proposto na especificação anterior: agora, os primeiros 64 bytes da mensagem carregarão o indicativo do tamanho da mensagem(representação em base 10), e o restante carrega a mensagem propriamente dita, de acordo com o diagrama a seguir:

Exemplo:

O corpo da mensagem deverá seguir o formato JSON, e poderá variar de acordo com o comando enviado pelo usuário. Deverá seguir a seguinte estrutura para cada comando:

- Comando do usuário: HELLO gadnlino
- Corpo da mensagem de requisição do cliente:

```
{
  "type":"HELLO",
  "data":{
    "username":"gadnlino"
  }
}
```

- Corpo da mensagem de resposta do servidor:


```
{
  "result":"gadnlino logged in successfully",
  "error":null
}
```


- Comando do usuário: BYE
- Corpo da mensagem de requisição do cliente:


```
{
  "type":"BYE",
  "data":null
}
```


- Corpo da mensagem de resposta do servidor:


```
{
  "result":"gadnlino logged off successfully",
  "error":null
}
```


- Comando do usuário: PEERS
- Corpo da mensagem de requisição do cliente:


```
{
  "type":"PEERS",
  "data":null
}
```


- Corpo da mensagem de resposta do servidor:


```
{
  "result": [
    {
      "username": "joaozinho123"
    },
    {
      "username": "mariazinha99"
    }
  ],
  "error": null
}
```


- Comando do usuário: MESSAGE joaozinho123 E ai joaozinhooo
- Corpo da mensagem de requisição do cliente:


```
{
  "type": "MESSAGE",
```

```
  "data": {
    "user": {
      "username": "joaozinho123"
    },
    "message_body": "E ai joaozinhooo",
    "timestamp": "1629057660"
  }
}
```

- Corpo da mensagem de resposta do servidor:

```
{
  "result": null,
  "error": "joaozinho123 is not logged in"
}
```

- Comando do usuário: LIST
- Corpo da mensagem de requisição do cliente:

```
{
  "type":"LIST",
  "data":null
}
```

- Corpo da mensagem de resposta do servidor:

```
{
  "result": [
    {
      "user": {
        "username": "gadnlino"
      },
      "message_body": "E ai joaozinhooo",
      "timestamp": "1629057660"
    }
  ],
  "error": null
}
```

- Comando do usuário: QUIT
- Corpo da mensagem de requisição do cliente:

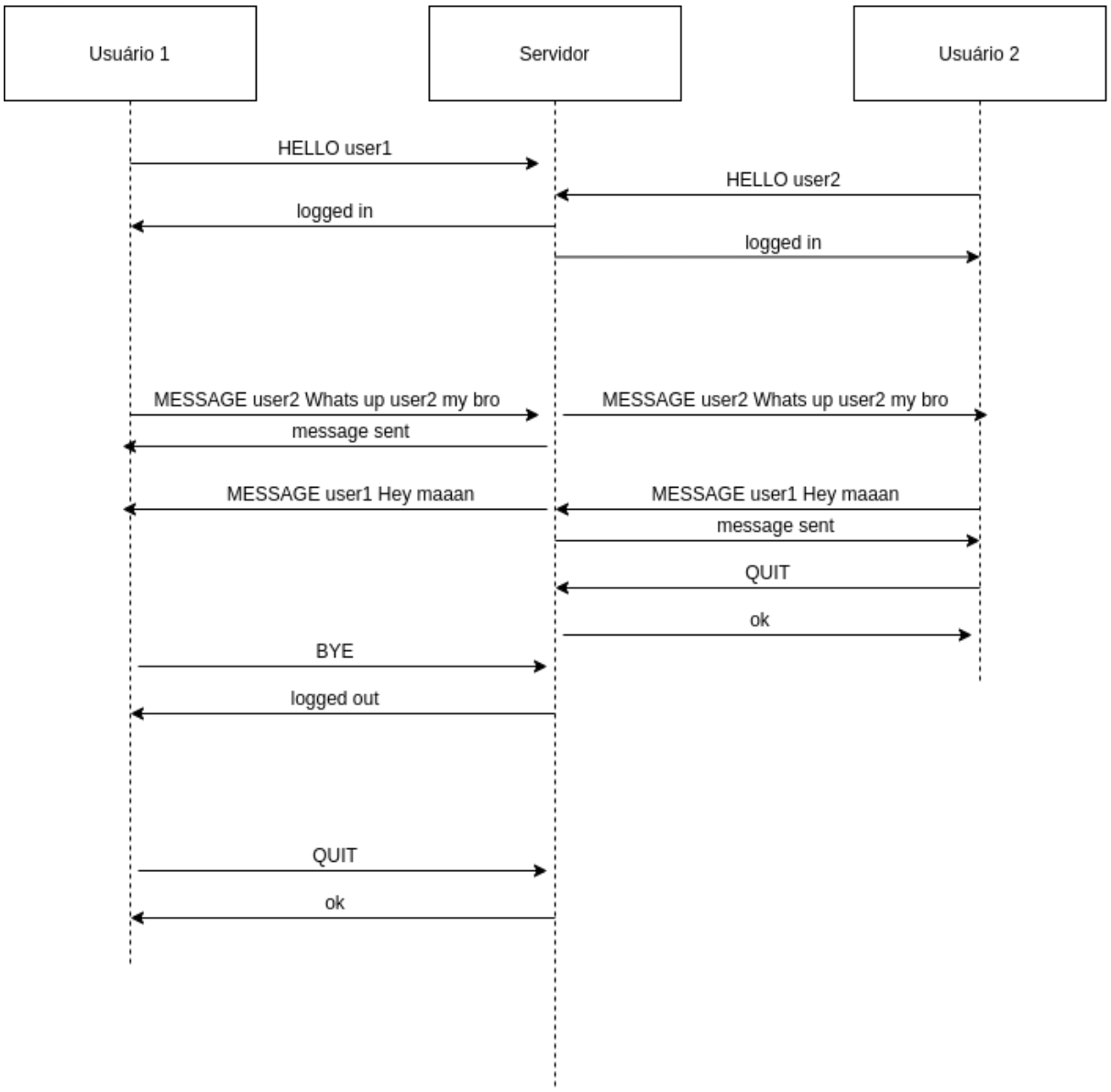
```
{
  "type": "QUIT",
  "data": null
}
```

- Corpo da mensagem de resposta do servidor:

```
{
```

```
  "result": "ok",
  "error": null
}
```

A fluxo de interação dos processos cliente/servidor se dá de acordo com o seguinte diagrama:



5- Recomendações para a execução local do projeto

O projeto apresenta dependências externas, fora da biblioteca padrão do python. Portanto, é necessário executar o seguinte comando no raiz do projeto:

```
pip3 install -r requirements.txt
```

Em seguida, deve-se iniciar o servidor:

python3 src/server.py

Por último, deve-se iniciar os clientes que enviarão entre si as mensagens(um em cada shell):

python src/client.py

Para interagir com os programas via linha de comando, deve-se utilizar a guia dos comandos de acordo com a seção 1.