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.

Seções:

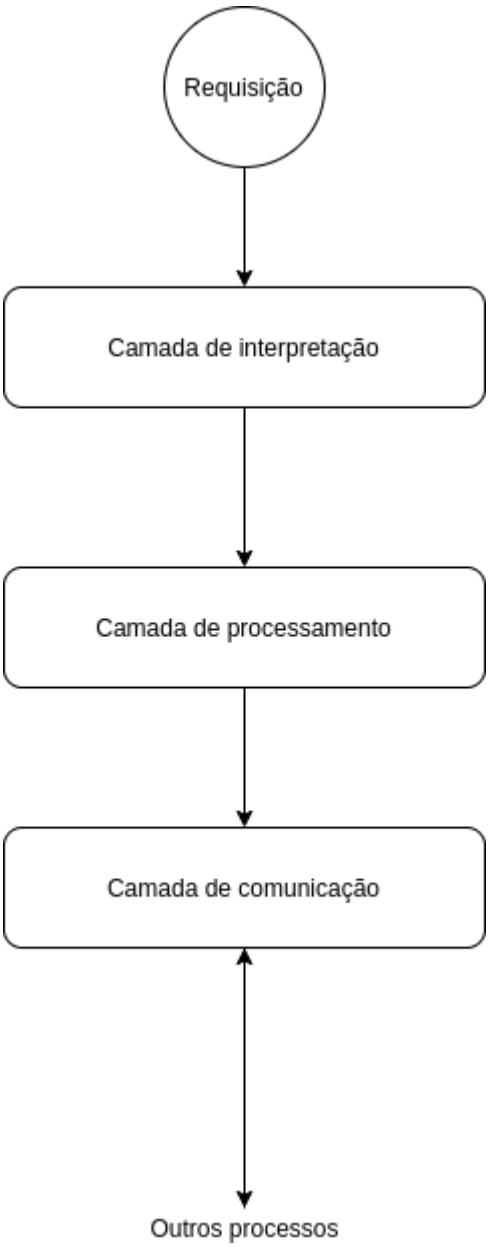
1 - Interface com o usuário

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

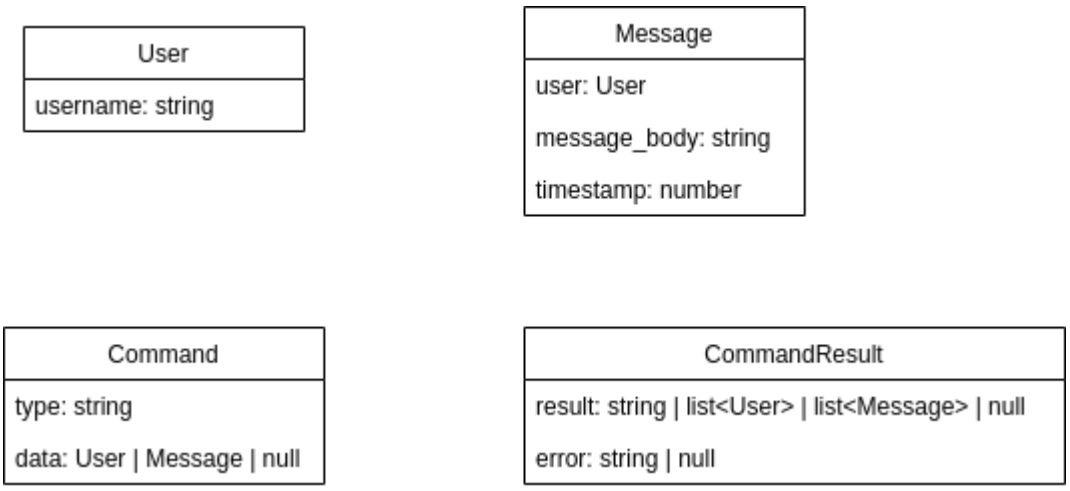
- **HELLO {username}**: Indica que o usuário está disponível para troca de mensagens.
- **BYE**: Torna o usuário indisponível para troca de mensagens.
- **PEERS**: Lista os demais usuários que se encontram disponíveis para troca de mensagens.
- **MESSAGE {username} {message_body}**: Envia uma mensagem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

2 - Arquitetura de software

A arquitetura será dividida em 3 camadas: a primeira camada será responsável por interpretar o comando do usuário e transformá-lo em um comando válido para ser trocado entre os processos,e, da mesma forma, receber a resposta da requisição e apresentá-la em um formato adequado para o usuário.
Logo abaixo, se encontra a camada de processamento, responsável por localizar o usuário destino para o envio de mensagens, e gerenciar o pool de usuários ativos e inativos.
Por último, existirá a camada de comunicação, que irá gerenciar o envio e o recebimento de mensagens pela rede. O diagrama a seguir exemplifica 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:



3 - Arquitetura de sistema

A aplicação será implementada na arquitetura cliente-servidor. Todos os processos deverão implementar as 3 camadas descritas acima, independentemente de serem clientes ou servidores. Particularmente nos clientes, a camada de processamento armazenará somente os dados dos usuários com quem esse cliente já iniciou uma conversa(ao contrário do servidor, que irá armazenar os dados de todos os usuários logados).

A interação entre esses 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 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.

Os primeiros 8 bytes da mensagem de requisição e de resposta deverão indicar o tamanho do corpo da mensagem, e o restante deverá conter o corpo da mensagem em si, como mostra a figura abaixo:

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
}
```

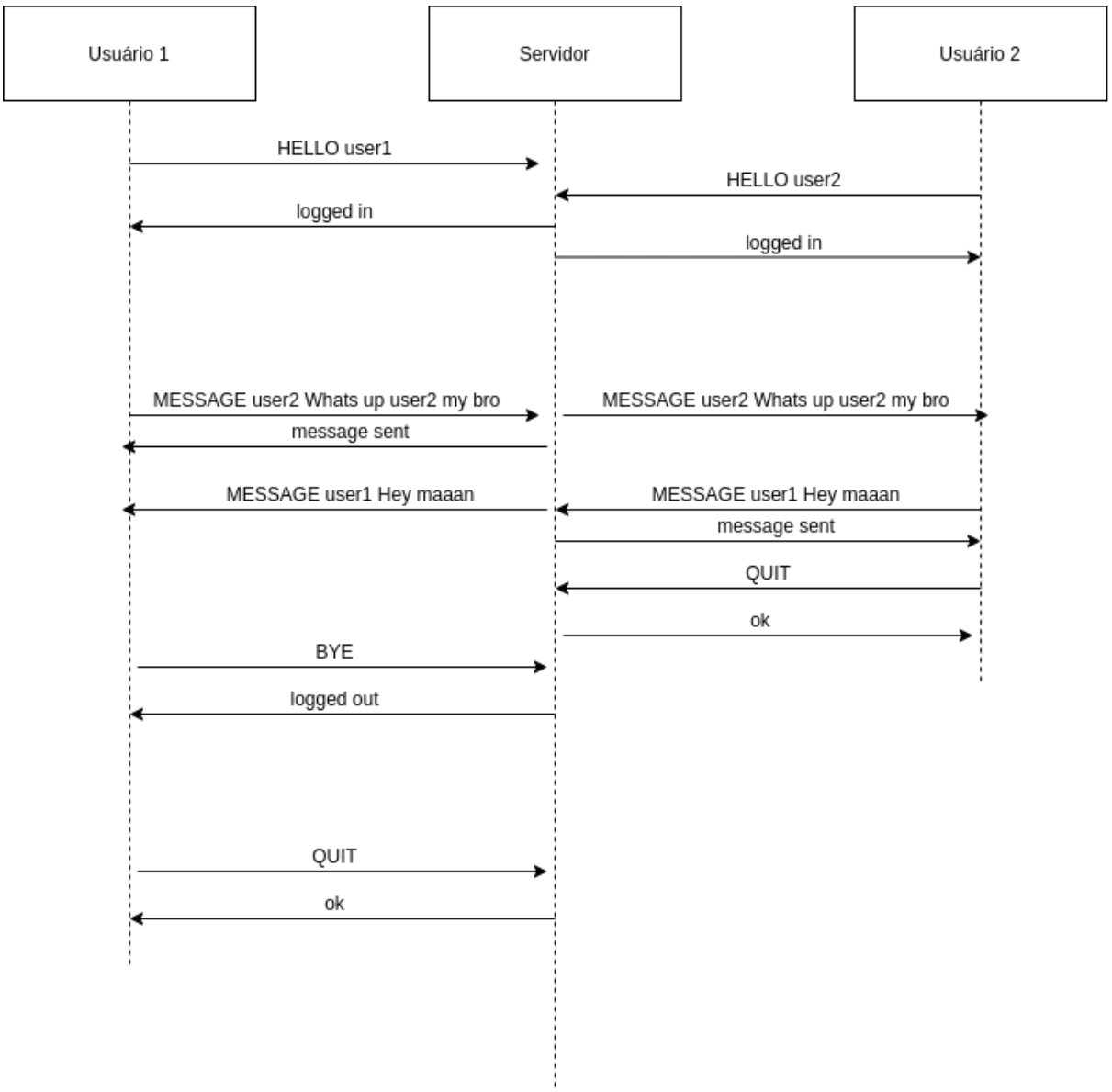
- Comando do usuário: QUIT
- Corpo da mensagem de requisição do cliente:

```
{
  "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:


Usuário 1
Servidor
Usuário 2
HELLO user1
HELLO user2
logged in
logged in
MESSAGE user2 Whats up user2 my bro
MESSAGE user2 Whats up user2 my bro
message sent
MESSAGE user1 Hey maaan
MESSAGE user1 Hey maaan
message sent
QUIT
ok
BYE
logged out
QUIT
ok