

# Laboratório 3

Joyce Brum

## Arquitetura de Software

Camadas:

1. Funcionalidades da **camada de interface** com o usuário: recebe do usuário o nome do arquivo de busca e exibe na tela o resultado do processamento. O resultado do processamento poderá ser:
  - a. uma mensagem de erro indicando que o arquivo não foi encontrado;
  - b. ou a lista de palavras com suas ocorrências.A camada já receberá as 10 palavras mais frequentes ordenadas por frequência e terá que imprimi-las.
2. Funcionalidades da **camada de processamento**: solicita o acesso ao arquivo texto. Se o arquivo for válido, realiza a contagem das palavras e prepara a resposta para ser devolvida para a camada de interface. Se o arquivo for inválido, responde com a mensagem de erro. A camada de processamento retornará para a camada de interface a lista com as 10 palavras ordenadas por frequência.
3. Funcionalidades da **camada de acesso aos dados**: verifica se o arquivo existe em sua base. Se sim, devolve o seu conteúdo inteiro. Caso contrário, envia uma mensagem de erro.

# Arquitetura de Sistema

Proposta de arquitetura de sistema:

1. **Lado cliente:** implementa a **camada de interface com o usuário**. O usuário poderá solicitar o processamento de um ou mais arquivos em uma única execução da aplicação: o programa espera pelo nome do arquivo, faz o processamento, retorna o resultado, e então aguarda um novo pedido de arquivo ou o comando de finalização.
2. **Lado servidor:** implementa a **camada de processamento** e a **camada de acesso aos dados**. O servidor será um **servidor concorrente**, isto é, que cria um fluxo de execução para cada cliente novo.

Formato das mensagens:

- A lista de palavras e frequências será representada como uma lista de pares no seguinte formato [(palavra1, frequencia1), (palavra2, frequencia2)].
- A camada de aplicação para solicitar o processamento de um arquivo para a camada de processamento deverá enviar uma **requisição** contendo o nome do arquivo em string a ser verificado e aguardará a **resposta** com a lista das 10 palavras mais frequentes do arquivo e suas frequências ou erro.
- A camada de processamento recebe a requisição com o nome do arquivo e envia uma **requisição** também contendo apenas o nome do arquivo à camada de acesso aos dados, aguardando a **resposta** com o conteúdo do arquivo ou erro.
- A camada de acesso aos dados recebe a requisição com o nome do arquivo, procura o arquivo na sua base de dados e retorna uma mensagem no formato de dicionário onde a chave "status" indica o status da busca (0 para sucesso e 1 para erro) e a chave "data" sendo o conteúdo do arquivo caso o mesmo seja encontrado ou "Arquivo não encontrado." caso o contrário.  
formato {'status': 0, 'data': "amor amor vida"} ou  
{'status': 1, 'data': "Arquivo não encontrado."}

## Comandos

### Cliente

Os comandos possíveis do lado do cliente são:

1. exit: encerra a aplicação

### Servidor - Processamento

Os comandos possíveis do lado do cliente são:

1. exit: espera até todos os clientes ativos encerrarem e encerra o servidor
2. hist: retorna a lista de todos os clientes que já se comunicaram

## Servidor - Acesso aos dados

1. exit: espera até todos os clientes ativos encerrarem e encerra o servidor
2. dir: lista todos os arquivos da base de dados no formato (IP, porta) ou retornando "Não houve conexão com nenhum cliente" no caso de não ter havido nenhuma conexão com cliente

## Exemplo execução

Dessa forma temos dois fluxos possíveis: um caso de sucesso e um caso de falha. Segue abaixo uma sequência de mensagens para cada fluxo. no caso de um arquivo de nome amo.txt que só tenha as palavras "amor amor vida"

	Cliente	Processamento	Acesso aos Dados
Sucesso	<pre>send-&gt; "amo.txt"  rcv &lt;- [(amor, 2), (vida, 1)]  imprime: amor - 2 vida - 1</pre>	<pre>rcv_client &lt;- "amo.txt"  send_access -&gt; "amo.txt"  rcv_access &lt;- "amor amor vida"  send_client -&gt; [(amor, 2), (vida, 1)]</pre>	<pre>rcv &lt;- "amo.txt"  send -&gt;{'status': 0, 'data': "amor amor vida"}</pre>
Falha	<pre>msg -&gt; amo.xlsx  rcv &lt;- "Arquivo não encontrado."  imprime: Arquivo não encontrado.</pre>	<pre>rcv_client &lt;- "amo.txt"  send_access -&gt; "amo.txt"  rcv_access &lt;- "Arquivo não encontrado."  send_client -&gt; "Arquivo não encontrado."</pre>	<pre>rcv &lt;- "amo.txt"  send -&gt; {'status': 1, 'data': "Arquivo não encontrado."}</pre>

# Análise

## Camada de acesso aos dados

O comando `dir` permite que o administrador obtenha de maneira fácil a lista de arquivos que estão na base de dados.

```
joyce@joyce-brum-dell:~/trabalhos-ufrj/Sistemas Distribuidos/Laboratorio 3$ python3 servidor-acesso-dados.py
Camada de acesso aos dados do servidor iniciada. Para encerra-la digite exit
dir
-----
lorem.txt
quod.txt
amo.txt
lab1.txt
lab2.txt
-----
█
```

O comando `exit` permite esta camada do servidor de maneira elegante.

```
joyce@joyce-brum-dell:~/trabalhos-ufrj/Sistemas Distribuidos/Laboratorio 3$ python3 servidor-acesso-dados.py
Camada de acesso aos dados do servidor iniciada. Para encerra-la digite exit
exit
Aguardando clientes terminarem para encerrar
Encerrando...
█
```

Se tiver algum cliente rodando ainda, o servidor aguardará que este termine antes de encerrar, e um log deixa isso transparente para o administrador.

```
joyce@joyce-brum-dell:~/trabalhos-ufrj/Sistemas Distribuidos/Laboratorio 3$ python3 servidor-acesso-dados.py
Camada de acesso aos dados do servidor iniciada. Para encerra-la digite exit
Procurando o arquivo lab1.txt.
Arquivo encontrado.

exit
Aguardando clientes terminarem para encerrar
█
```

Assim que o cliente encerra, o servidor imprime `Encerrando...` e encerra.

```
joyce@joyce-brum-dell:~/trabalhos-ufrj/Sistemas Distribuidos/Laboratorio 3$ python3 servidor-acesso-dados.py
Camada de acesso aos dados do servidor iniciada. Para encerra-la digite exit
Procurando o arquivo lab1.txt.
Arquivo encontrado.

exit
Aguardando clientes terminarem para encerrar
Encerrando...
█
```

## Camada de processamento

O comando `'hist'` permite que o administrador obtenha um log completo de todos os endereços e portas dos clientes que acessaram o servidor.

Caso não tenha tido nenhuma conexão ainda:

```
joyce@joyce-brum-dell:~/trabalhos-ufrj/Sistemas Distribuidos/Laboratorio 3$ python3 servidor-processamento.py
Camada de processamento do servidor iniciada. Para encerra-la digite exit
hist
-----
Não houve conexão com nenhum cliente
-----
█
```

Após três conexões:

```
joyce@joyce-brum-dell:~/trabalhos-ufrj/Sistemas Distribuidos/Laboratorio 3$ python3 servidor-processamento.py
Camada de processamento do servidor iniciada. Para encerra-la digite exit
hist
-----
Não houve conexão com nenhum cliente
-----
Conectado com: ('127.0.0.1', 49160)
Conectado com: ('127.0.0.1', 49164)
Conectado com: ('127.0.0.1', 49168)
hist
-----
('127.0.0.1', 49160)
('127.0.0.1', 49164)
('127.0.0.1', 49168)
-----
```

O comando exit tem um comportamento parecido com o da camada de acesso aos dados. Caso não tenha nenhum cliente conectado ele só encerra.

```
joyce@joyce-brum-dell:~/trabalhos-ufrj/Sistemas Distribuidos/Laboratorio 3$ python3 servidor-processamento.py
Camada de processamento do servidor iniciada. Para encerra-la digite exit
exit
Aguardando clientes terminarem para encerrar
Encerrando...
```

Caso ainda tenha algum cliente, a mensagem “Aguardando clientes terminarem para encerrar” informará ao administrador que a camada está aguardando para poder encerrar.

```
joyce@joyce-brum-dell:~/trabalhos-ufrj/Sistemas Distribuidos/Laboratorio 3$ python3 servidor-processamento.py
Camada de processamento do servidor iniciada. Para encerra-la digite exit
Conectado com: ('127.0.0.1', 49206)
Conectado com: ('127.0.0.1', 49210)
exit
Aguardando clientes terminarem para encerrar
█
```

Assim que os processos encerram, ela termina.

```
joyce@joyce-brum-dell:~/trabalhos-ufrj/Sistemas Distribuidos/Laboratorio 3$ python3 servidor-processamento.py
Camada de processamento do servidor iniciada. Para encerra-la digite exit
Conectado com: ('127.0.0.1', 49206)
Conectado com: ('127.0.0.1', 49210)
exit
Aguardando clientes terminarem para encerrar
Encerrando...
```

## Camada de interface com o usuário

A camada de interface manteve o mesmo comportamento desenvolvido no Laboratório 2.

O comando exit encerrando a aplicação.

```
joyce@joyce-brum-dell:~/trabalhos-ufrj/Sistemas Distribuidos/Laboratorio 3$ python3 cliente.py
Para encerrar digite 'exit'
Digite o nome do arquivo que deseja processar
exit
```

Enquanto qualquer outro comando é interpretado como um nome de arquivo a ser processado.

```

joyce@joyce-brum-dell:~/trabalhos-ufrrj/Sistemas Distribuidos/Laboratorio 3$ python3 cliente.py
Para encerrar digite 'exit'
Digite o nome do arquivo que deseja processar
hist
Arquivo não encontrado.

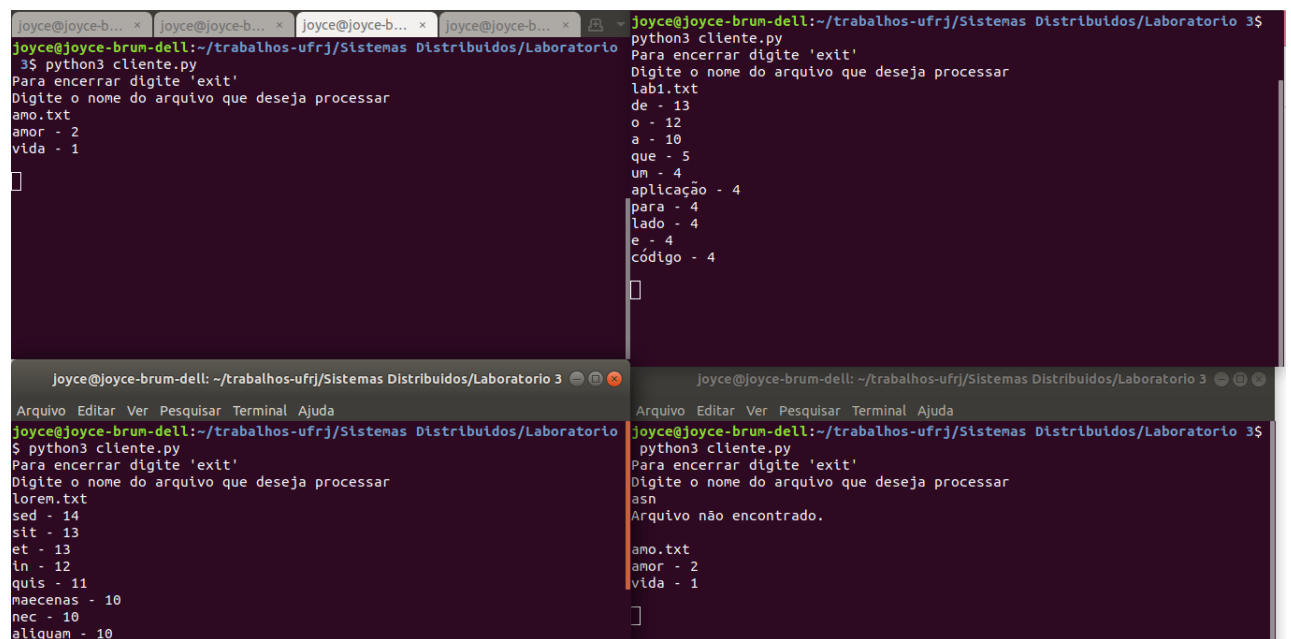
dir
Arquivo não encontrado.

amo.txt
amor - 2
vida - 1

lab1.txt
de - 13
o - 12
a - 10
que - 5
um - 4
aplicação - 4
para - 4
lado - 4
e - 4
código - 4

```

Além disso agora é possível ter mais de um cliente consultando ao mesmo tempo, pois para cada cliente é criado um fluxo de execução distinto



```

joyce@joyce-brum-dell:~/trabalhos-ufrrj/Sistemas Distribuidos/Laboratorio 3$ python3 cliente.py
Para encerrar digite 'exit'
Digite o nome do arquivo que deseja processar
amo.txt
amor - 2
vida - 1

joyce@joyce-brum-dell:~/trabalhos-ufrrj/Sistemas Distribuidos/Laboratorio 3$ python3 cliente.py
Para encerrar digite 'exit'
Digite o nome do arquivo que deseja processar
lab1.txt
de - 13
o - 12
a - 10
que - 5
um - 4
aplicação - 4
para - 4
lado - 4
e - 4
código - 4

joyce@joyce-brum-dell:~/trabalhos-ufrrj/Sistemas Distribuidos/Laboratorio 3$ python3 cliente.py
Para encerrar digite 'exit'
Digite o nome do arquivo que deseja processar
lorem.txt
sed - 14
sit - 13
et - 13
in - 12
quis - 11
maecenas - 10
nec - 10
aliquam - 10

joyce@joyce-brum-dell:~/trabalhos-ufrrj/Sistemas Distribuidos/Laboratorio 3$ python3 cliente.py
Para encerrar digite 'exit'
Digite o nome do arquivo que deseja processar
asn
Arquivo não encontrado.

joyce@joyce-brum-dell:~/trabalhos-ufrrj/Sistemas Distribuidos/Laboratorio 3$ python3 cliente.py
Para encerrar digite 'exit'
Digite o nome do arquivo que deseja processar
amo.txt
amor - 2
vida - 1

```