Universidade Estadual de Mato Grosso do Sul

Curso de Ciência da Computação

Disciplina de Redes de Computadores

Trabalho de Redes de Computadores

PLATAFORMA DE MONITORAMENTO INTELIGENTE DE FILAS EM TEMPO REAL

Dourados (MS), 22 de maio de 2026.

INTRODUÇÃO

Desenvolvimento de uma plataforma cliente-servidor para monitoramento e gerenciamento inteligente de filas em tempo real, utilizando múltiplos clientes concorrentes conectados simultaneamente ao servidor central.
O sistema permitirá:

- Cadastro e autenticação de usuários;
- Atualização dinâmica de filas de atendimento;
- Notificações em tempo real;
- Painéis administrativos;
- Comunicação simultânea entre centenas de clientes;
- Balanceamento de carga e controle de concorrência.

Diversos locais que atendem a comunidade frequentemente sofrem com:

- superlotação;
- filas desorganizadas;
- ausência de previsibilidade;
- perda de tempo no atendimento;
- dificuldade de comunicação entre setores;
- falta de atualização em tempo real para pacientes.

O projeto propõe um sistema onde:

- usuários acompanham sua posição na fila via aplicativo/web;
- recepcionistas atualizam atendimentos;
- coordenadores recebem novos usuários automaticamente;
- administradores monitoram métricas em tempo real.

OBJETIVO GERAL

Construir um sistema de redes baseado em arquitetura cliente-servidor com suporte a múltiplos clientes concorrentes utilizando técnicas de comunicação em rede.

TÉCNICAS DE MÚLTIPLOS CLIENTES

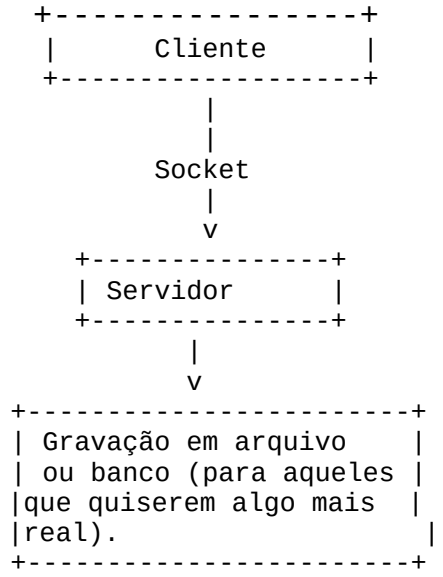
O servidor deverá suportar múltiplas conexões simultâneas utilizando uma das técnicas:

- Multiplexação;
- Processos Fork();
- Multithreads

O servidor principal deverá ficar responsável apenas por:

- Aceitar conexões;
- Distribuir eventos;
- Monitorar sockets;

ARQUITETURA DO SISTEMA



Responsabilidades dos Clientes:
- enviar solicitações;
- receber notificações;
- atualizar filas em tempo real;
- autenticar usuários;

Responsabilidades do Servidor:
- autenticação;
- gerenciamento de filas;
- sincronizar os clientes;
- envio de notificações;

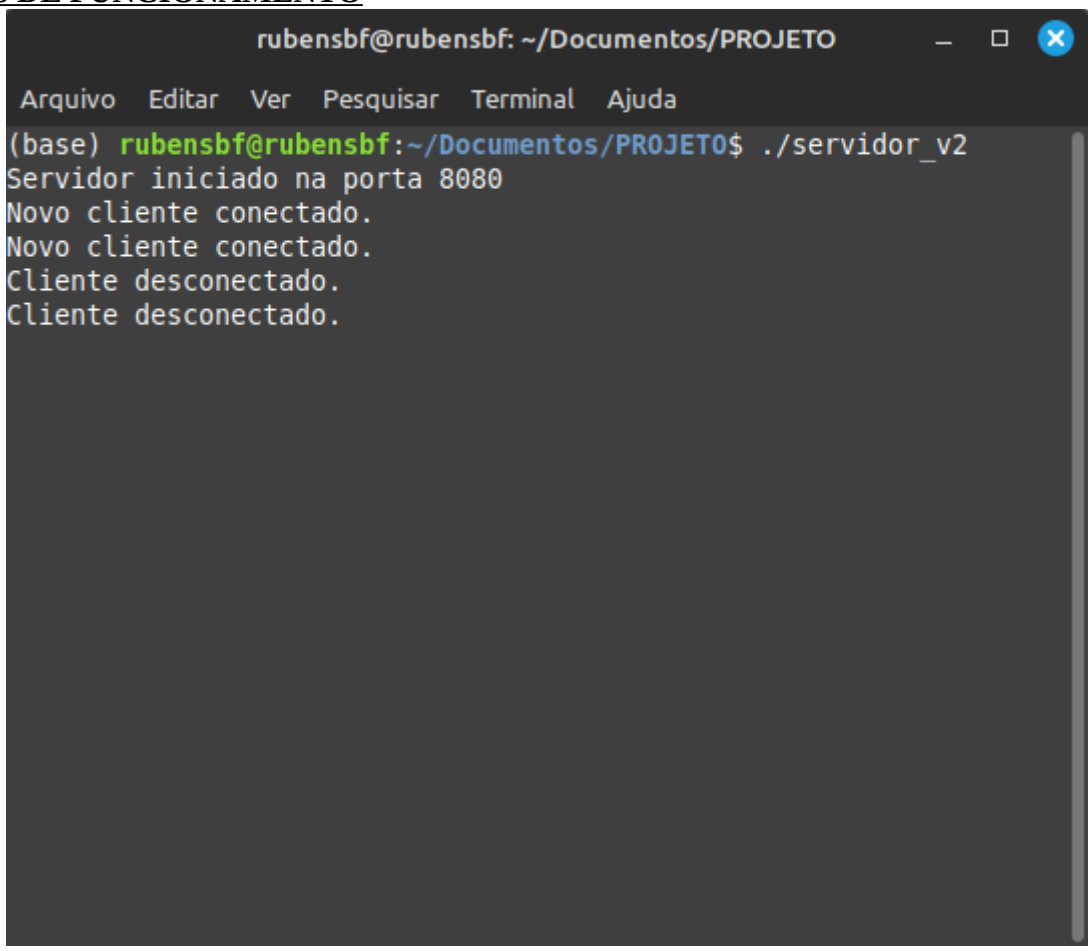
Gravação em Arquivo / Banco de Dados (não obrigatório).
Dados a serem armazenados:
- usuários;
- filas;
- histórico;
- logs;
- sessões (dados do cliente que logou);

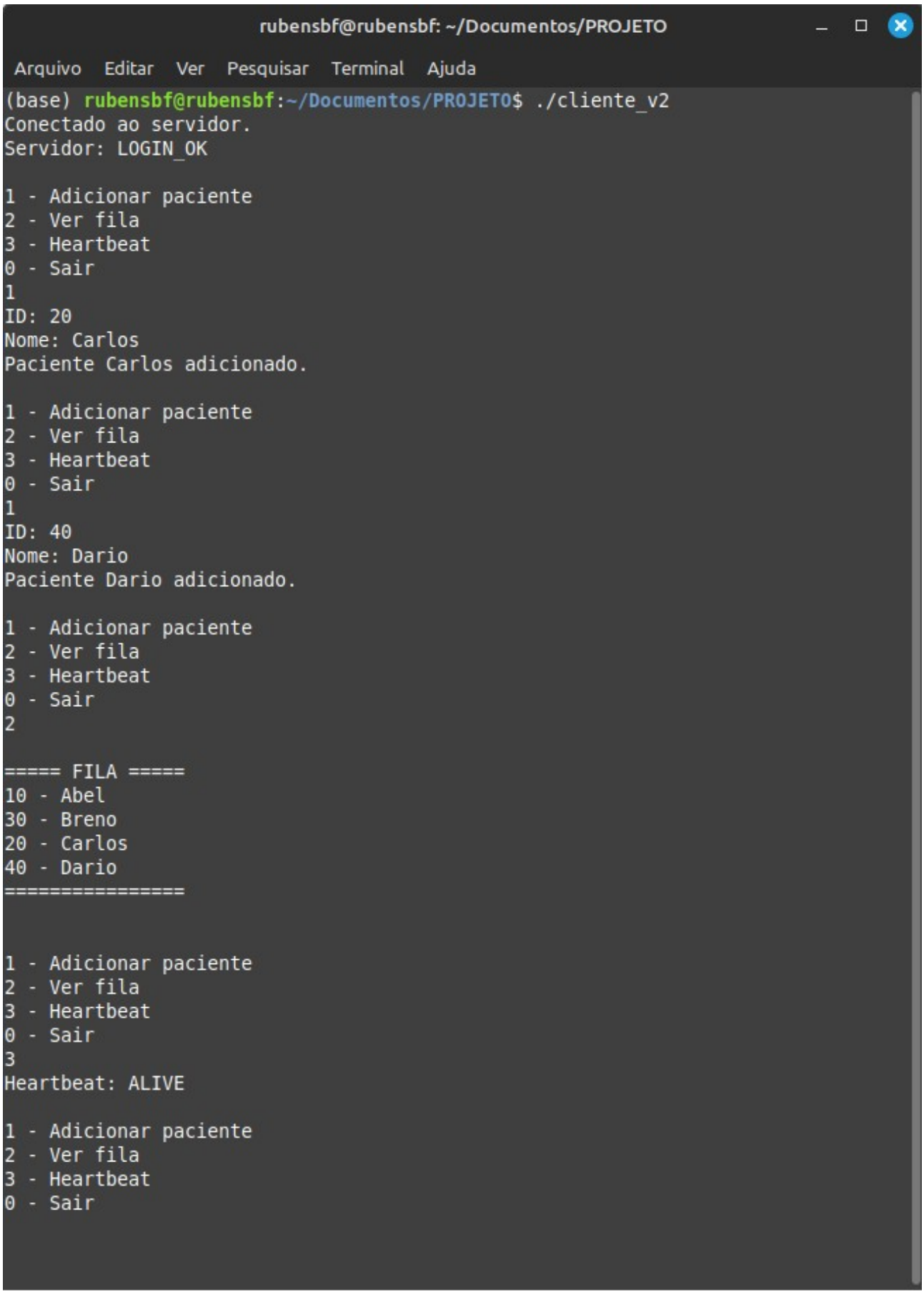
SISTEMAS

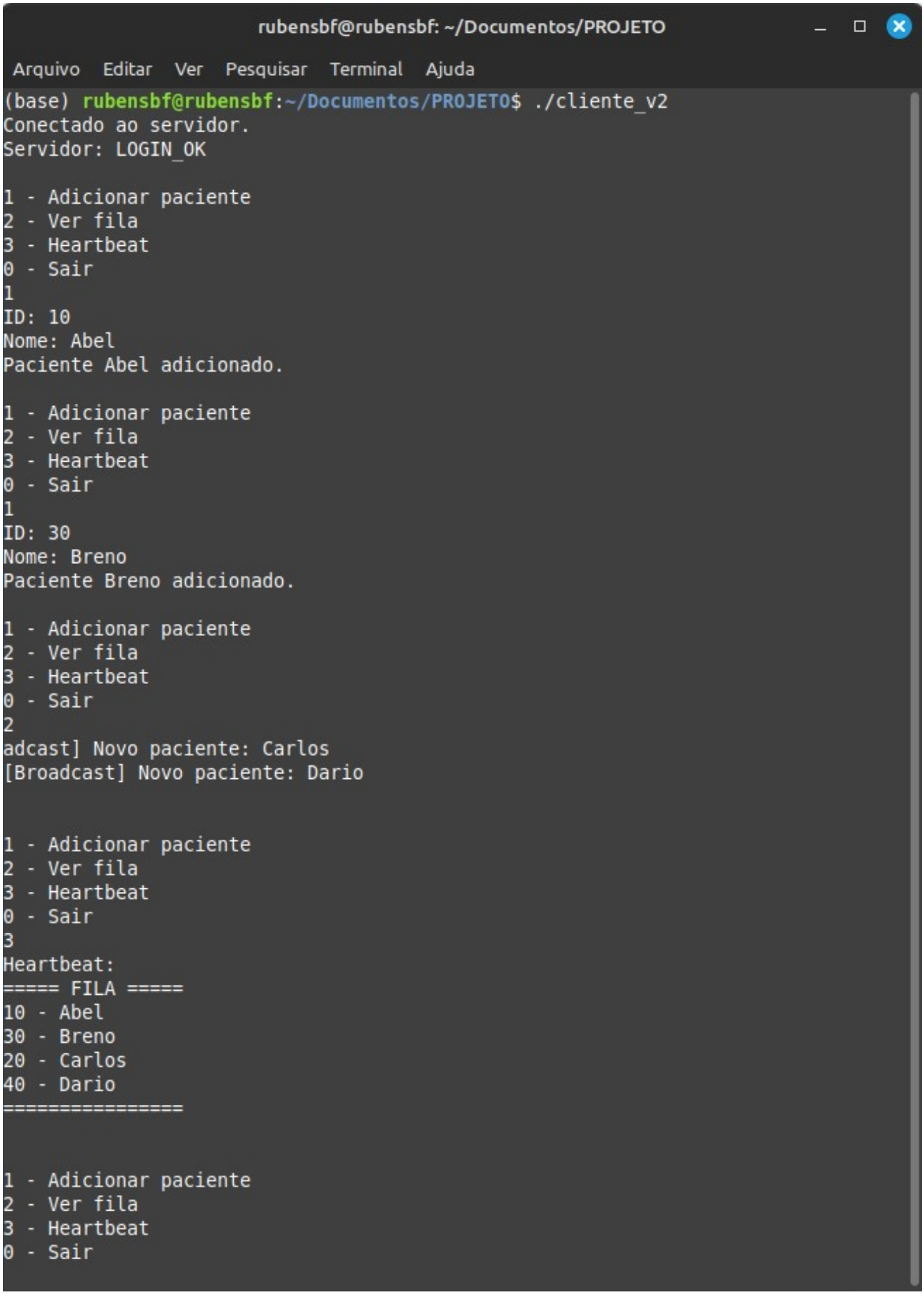
Tanto o cliente quanto o servidor são inicializados da seguinte forma:
- ./cliente
- ./servidor

O programa cliente se conectará ao servidor por meio do endereço IP e porta de forma automática, sem necessidade de parâmetros.

TELAS DE FUNCIONAMENTO




rubensbf@rubensbf: ~/Documentos/PROJETO
Arquivo Editar Ver Pesquisar Terminal Ajuda
(base) rubensbf@rubensbf:~/Documentos/PROJETO$ ./cliente_v2
Conectado ao servidor.
Servidor: LOGIN_OK
1 - Adicionar paciente
2 - Ver fila
3 - Heartbeat
0 - Sair
1
ID: 20
Nome: Carlos
Paciente Carlos adicionado.
1 - Adicionar paciente
2 - Ver fila
3 - Heartbeat
0 - Sair
1
ID: 40
Nome: Dario
Paciente Dario adicionado.
1 - Adicionar paciente
2 - Ver fila
3 - Heartbeat
0 - Sair
2
===== FILA =====
10 - Abel
30 - Breno
20 - Carlos
40 - Dario
1 - Adicionar paciente
2 - Ver fila
3 - Heartbeat
0 - Sair
3
Heartbeat: ALIVE
1 - Adicionar paciente
2 - Ver fila
3 - Heartbeat
0 - Sair


rubensbf@rubensbf: ~/Documentos/PROJETO
Arquivo Editar Ver Pesquisar Terminal Ajuda
(base) rubensbf@rubensbf:~/Documentos/PROJETO$ ./cliente_v2
Conectado ao servidor.
Servidor: LOGIN_OK
1 - Adicionar paciente
2 - Ver fila
3 - Heartbeat
0 - Sair
1
ID: 10
Nome: Abel
Paciente Abel adicionado.
1 - Adicionar paciente
2 - Ver fila
3 - Heartbeat
0 - Sair
1
ID: 30
Nome: Breno
Paciente Breno adicionado.
1 - Adicionar paciente
2 - Ver fila
3 - Heartbeat
0 - Sair
2
adcast] Novo paciente: Carlos
[Broadcast] Novo paciente: Dario
1 - Adicionar paciente
2 - Ver fila
3 - Heartbeat
0 - Sair
3
Heartbeat:
===== FILA =====
10 - Abel
30 - Breno
20 - Carlos
40 - Dario
1 - Adicionar paciente
2 - Ver fila
3 - Heartbeat
0 - Sair

Observação: A função *Heartbeat* deve devolver ao cliente a lista de usuários cadastrados por outros clientes. Caso não haja novas inserções, a função deve devolver o termo **ALIVE**.

ENTREGA DO CÓDIGO

O código e a documentação devem ser entregues em um arquivo Zip. Dentro deste arquivo Zip devem conter um **readme.txt** com o nome do aluno e o comando para a execução do código e, um arquivo PDF da documentação. Inclua todos os arquivos fontes (.c, .h, makefile, **não incluam executáveis ou arquivos objetos**), em um único diretório. Um **Makefile** deve ser fornecido para a compilação do código.

Parte desse trabalho envolve o aprendizado de como construir um makefile e utilizar a ferramenta make. Este makefile, quando executado sem parâmetros, irá gerar os dois programas: cliente e servidor, EXATAMENTE com esses nomes.

Submissões onde os programas não sigam as especificações de parâmetro e nomes, makefiles não funcionando ou arquivos necessários faltando **NÃO SERÃO CORRIGIDOS.** Programas que não compilem também não serão corrigidos.

DOCUMENTAÇÃO

O texto da documentação deve ser breve, de forma que o professor possa entender o que foi feito no código sem ter que entender linha a linha dos arquivos. Implementações modularizadas deverão mencionar quais funções são implementadas em cada módulo ou classe. A documentação deverá conter os seguintes itens:

1. O sumário do problema a ser tratado;
2. Uma descrição sucinta dos algoritmos e dos tipos abstratos de dados, das principais funções, procedimentos e das decisões de implementação;
3. Decisões de implementação que porventura estejam omissos na especificação;
4. Como foi tratada a retransmissão de mensagens;
5. Testes, mostrando que o programa está funcionando de acordo com as especificações, seguidos de sua análise;
6. Testar o código com uma carga de 100 clientes, 1000 clientes e 10000 clientes. Essa geração deverá ser automática (nós fizemos esse tipo de teste nas aulas de lab). Colocar o print screen de forma que seja possível identificar todas as conexões. Se achar mais prático pode-se criar um segundo código cliente com os parâmetros de entrada 100, 1000, 10000.
7. Print screens mostrando o correto funcionamento do cliente e do servidor e exemplos de testes executados;
8. Conclusão e referências bibliográficas.

AVALIAÇÃO

A avaliação do trabalho será composta pela execução dos programas desenvolvidos e pela análise da documentação. Todos os alunos deverão apresentar e explicar em laboratório o trabalho.

Os seguintes itens serão avaliados:

- A qualidade do código (código bem organizado, estruturado, com comentários explicativos, variáveis com nomes intuitivos, modularidade, etc);
- Execução correta do código com entrada de testes a serem definidas no momento da avaliação. As entradas de testes irão exercitar a funcionalidade completa do código e, testar casos especiais ou de maior dificuldade de implementação, mas que devem ser tratados por um programa correto;
- A aderência ao protocolo especificado. Usarei ferramentas de captura de tráfego da rede (wireshark) e analisarei o código para verificar se a comunicação está da forma definida na documentação;
- Conteúdo da documentação que deve conter os itens mencionados anteriormente;

- Coerência e coesão da documentação (apresentação visual e organização, uso correto da língua portuguesa, qualidade textual e facilidade de compreensão);
- CASOS DE CÓPIA NÃO SERÃO TOLERADOS;
- Qualquer elemento que não esteja especificado neste documento, mas que tenha que ser inserido para que o protocolo funcione, deve ser descrito na documentação de forma explícita.

REFERÊNCIA

Para uso do programa Make e escrita de Makefiles:
http://www.gnu.org/software/make/manual/make.html
http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/

ENTREGA E APRESENTAÇÃO DO TRABALHO

Data: 29/07/2026
O trabalho deverá ser feito individualmente.