Desenvolvimento de uma Aplicação Cliente-Servidor de Urna Eletrônica Utilizando Socket TCP.

Anderson Valesan[1] **8599987**
Danilo Françoso Tedeschi[1] **8937361**
Yuri Tavares[1] **8936499**

[1]Instituto de Ciências Matemáticas e Computação - Universidade de São Paulo

Sumário

1 Introdução **3**

2 Descrição do Problema **3**

3 Fundamentação Teórica **4**

4 Detalhamento da Solução **4**

4.1 Metodologia, ambientes e ferramentas Utilizadas 4

4.2 Modelagem do sistema a ser desenvolvido/estudado 4

4.3 Funcionamento 5

5 Conclusão **5**

6 Figuras **7**

1. Introdução

Com o conhecimento adquirido na disciplina de Redes e alguns conhecimentos adquiridos na disciplina de Engenharia de Software, foi decidido por meio desse trabalho descrever a metodologia empregada, tanto na formulação mais generalizada de resolução do problema, utilizando para isto diagramas conceituais que nos dão uma visão mais ampla do problema, facilitando sua resolução, até as partes mais específicas, onde, de fato, o problema é resolvido utilizando clusters como apoio para armazenamento de resultados. Assim como os passos e análises pensados pelo grupo para se chegar ao resultado final respondendo ao desafio de construir, apenas com a descrição do problema, uma urna eletrônica que computa votos e mostra candidatos conectando em um cluster por meio de sockets.

2. Descrição do Problema

O problema consiste no desenvolvimento e planejamento de uma aplicação cliente-servidor de uma urna eletrônica usando linguagem Java e utilizando socket(TCP), levando em consideração os requisitos listados abaixo:

- Ser implementado exclusivamente na Linguagem Java;
- Utilizar sockets;
- Utilize uma classe "Candidato" que contenha os seguintes atributos:
  - int código_votação
  - String nome_candidato
  - String partido
  - int num_votos
- O lado cliente deve possuir uma interface (gráfica ou não) de votação com os seguintes items:
  - 1 Votar (Irá solicitar o código de votação de um candidato)
  - 2 Votar branco
  - 3 Votar nulo
  - 999 Carregar a lista de candidatos do servidor (Nesta opção deve-se enviar uma mensagem ao servidor com um código (ex.: 999) que irá direcionar qual operação será realizada
  - 888 Finalizar as votações da urna e enviar ao servidor

Além de certas restrições que competem as votações:

- A urna só permitirá votações caso a lista de candidatos já tenha sido carregada do servidor
- A urna só permitirá enviar os resultados da urna ao servidor, caso haja no mínimo uma votação realizada

E restrições que competem a interface do lado do servidor:

- Abrir uma thread para controlar novas conexões, em que cada conexão poderá variar de acordo com um "opcode" vindo do cliente:
  - "opcode == 999" - Abre uma conexão que irá enviar a lista de candidatos ao cliente
  - "opcode == 888" - Abre uma conexão que irá receber os votos de cada candidato da urna
- Após o início da thread na função "main", recomenda-se utilizar um laço de repetição infinito para apresentar as parciais de votos computados de cada candidato

3. Fundamentação Teórica

Para melhor entendimento do problema proposto e elaboração de uma resposta mais eficiente, foi necessário o aprofundamento de certos conceitos, como Thread e Multithread, é uma forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrentemente , socket, que é uma ligação entre dois processos para que ambos possam se comunicar, [Tanenbaum 2010]Diagrama de Caso de Uso, Diagrama de Sequência, Diagrama de Comunicação, que são diagramas úteis para modelar o projeto e o entender, antes de de fato o implementar [PRESSMAN and MAXIM 2014].

4. Detalhamento da Solução

4.1. Metodologia, ambientes e ferramentas Utilizadas

Assim como solicitado, foi utilizada a linguagem de programação Java e o ambiente de desenvolvimento Netbeans para o desenvolvimento completo e estruturado do projeto, foi-se também utilizada o programa Astah para se implementar os diagramas em UML utilizados para representar um melhor entendimento do problema além de facilitar a visão do problema como um todo e utilizado LaTeXpara formatação do relatório.

4.2. Modelagem do sistema a ser desenvolvido/estudado

O sistema como um todo foi modelado seguindo os passos de diagramação de um sistema convencional, começando com um diagrama de casos de uso(Figura: 1), onde as funcionalidades principais e mais gerais do sistema foram organizadas, levando em consideração quem serão os atores daquele sistema, quais funções cada ator pode fazer e em que situação, além de como eles se relacionam entre si. Neste caso a Urna tem a funcionalidade de efetuar o voto e o servidor de contabilizar e consultar votos e de consultar candidatos.

Em seguida, foi feito um diagrama de sequência(Figura: 2), que representa a sequência lógica de forma simples da utilização do sistema por um eleitor e quais são os passos e a ordem em que o sistema os efetuará desde a chegada do eleitor até o término e computo dos votos, passando este para o sistema realizar o processo.

O diagrama de comunicação, representado na figura 3, nos traz informações importantes e de maneira um pouco mais detalhada de como a comunicação é efetuada e quais dados são passados e relacionados entre os atores.

Na figura 4 foi-se pensado em uma máquina de estado finito que corresponde a todas as funções com foco no sistema e como ele irá se comportar diante das diversas situações apresentadas, primeiramente o sistema está ocioso até a chegada de um eleitor, este então começa a apertar os dígitos, caso seja “999”, a lista de candidatos é atualizada, ele então escolhe um candidato efetuando o voto, quando os votos terminam, então se aperta “888” para se computar os votos no servidor.

4.3. Funcionamento

Link github: https://github.com/andersonvalesan/redes

- Executando o Servidor
  - Executar o programa PuTTY com os seguintes parâmetros:
    Host Name: cosmos.lasdpc.icmc.usp.br
    Port: 22200
  - Executar login como: grupo01bsi e senha: Ch5besas
  - Executar o comando: ssh node10
  - Executar o comando: java –jar Servidor.jar (uma vez que este já esteja no local)
- Executando o Cliente
  - Abrir o prompt de comando
  - Ir ao diretório do Cliente
  - Executar o comando: java –jar Cliente.jar
- Operações:
  As operações são realizadas de acordo com os seguintes opCodes:
  1: Voto em candidato, abrangendo-se nos números dos candidatos, que são: 13666, 24240, 44444, 30303
  2: Voto em branco
  3: Voto nulo
  888: Finalizar votação da urna
  999: Carregar lista de candidatos na urna
- Código:
  O código provê utilização de vários usuários votando ao mesmo tempo, ou seja, múltiplas urnas conectadas com o servidor. Também é capaz de salvar as votações em um arquivo para consulta posterior.
- Explicando o Código Várias funções explicam como comentário o que elas fazem, porém destacamos alguns pontos importantes:
  - novaListaCandidato() Cria-se uma nova lista de candidatos no servidor, porém apenas quando não houver nenhum arquivo de lista no diretório para ser carregada.
  - verificaCandidato() Verifica constantemente os números digitados, para que quando haja algum candidato com tal número sua foto aparecerá na urna.
  - tratamentoDeTelas() Faz o tratamento da interface, ajustando as telas de acordo com a situação;

  Vale ressaltar que tentamos modularizar mais, porém a reutilização de função seria escassa, senão nenhuma.

5. Conclusão

Com a modelagem do nosso projeto, todos os integrantes conseguiram efetuar contato com a programação de interfaces via Swing, que foi algo muito interessante para o aprendizado, pois através dele conseguimos montar a urna e descobrir suas vantagens e limitações, como a facilidade de realizar sua construção, porém com funções que não são modificáveis através do código.

Por meio deste trabalho pudemos nos aprofundar mais em relação tanto ao conteúdo passado quanto a modelagem de um projeto em si, em relação ao conteúdo, foi usado tcp por permitir uma maior confiabilidade que foi um quesito muito importante na construção da urna e um particular desafio foi a implementação de multithread, tendo que passar o socket como parâmetro da função

Referências

PRESSMAN, R. S. and MAXIM, B. (2014). Software engineering: An practitioner's approach. In Education, M.-H., editor, *Software Engineering: An Practitioner's Approach*.

Tanenbaum, A. S. (2010). Sistemas operacionais modernos. In Pearson, editor, *Sistemas Operacionais Modernos*.

6. Figuras

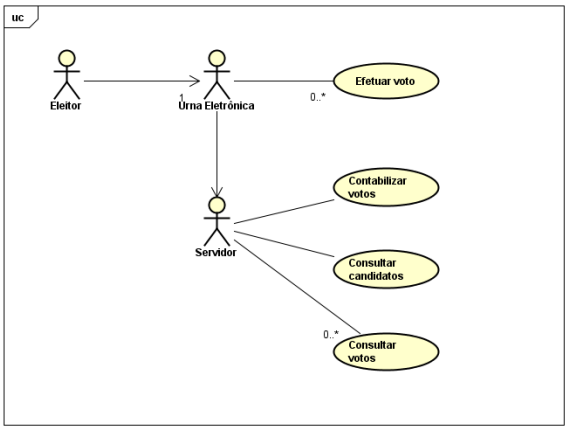

Figura 1. Diagrama de Caso de Uso

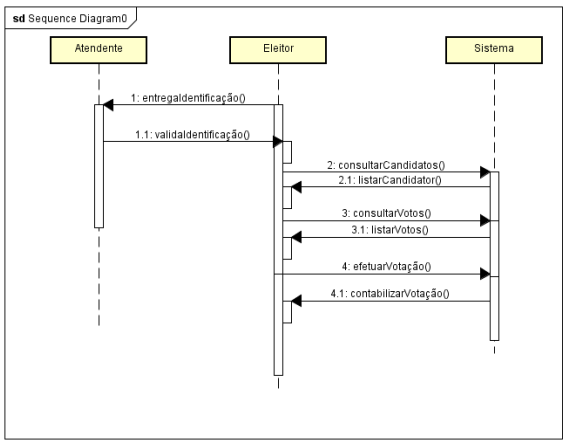

Figura 2. Diagrama de Sequência

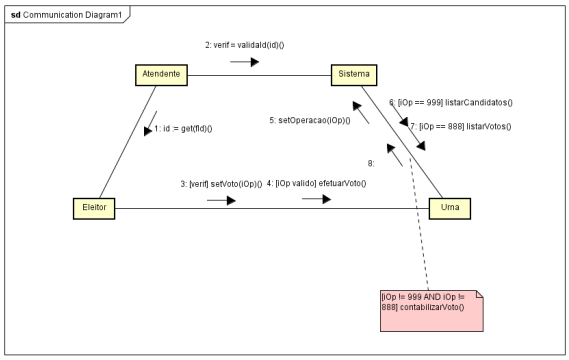

Figura 3. Diagrama de Comunicação

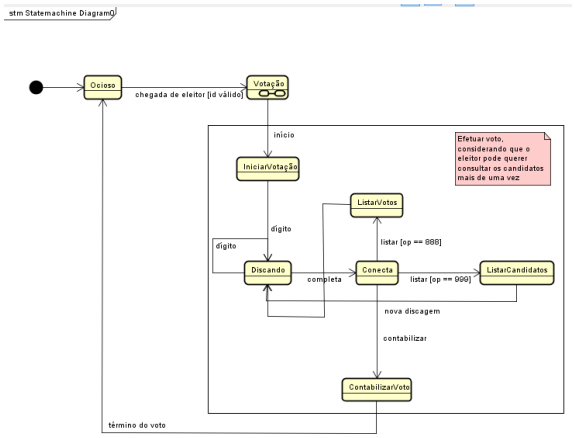

Figura 4. Diagrama de Máquina de Estado