8º FÓRUM ENSINO · PESQUISA EXTENSÃO · GESTÃO FEPEG
UNIVERSIDADE: SABERES E PRÁTICAS INOVADORAS
Trabalhos científicos · Apresentações artísticas e culturais · Debates · Minicursos e Palestras
REALIZAÇÃO: Unimontes
APOIO: FAPEMIG FADENOR
24 a 27 setembro
Campus Universitário Professor Darcy Ribeiro
www.fepeg.unimontes.br

DESENVOLVIMENTO DE UM SISTEMA DISTRIBUÍDO UTILIZANDO SOCKET

Raimundo Sebastião Teodoro Santana, Rodrigo Everton Soares Oliveira, Gabriela Santos Silva, Diorgenes Ferreira, Hiorrana Saely Silva Dias, Hugo Luciano de Souza Leal, Marcos Vinicius Soares Santos

Introdução

Ao evoluir do hardware e do software, foi importante prover ao usuário a total transparência do que se realizava enquanto este buscava, inseria, alterava ou compartilhava informações, não compelindo ao seu conhecimento - Cliente - saber o contexto/arquitetura no qual o sistema estava disposto. Nesse contexto, os sistemas distribuídos ganharam importância e se tornaram uma grande ferramenta de comunicação, troca de recursos e sincronização de dados, não tendo a distância como um fator a atrapalhar. Assim, surge a necessidade constante de disseminação de informação e comunicação. Segundo CARVALHO [2], as organizações buscaram ferramentas efetivas de comunicação, prezando que assuntos da organização não saíssem do domínio das mesmas. Deram-se o nome a esse meio de "ba", onde o conhecimento é criado, compartilhado e utilizado. E esses tipos de aplicações distribuídas estão se tornando cada vez mais comum devido à, principalmente, a popularização das redes de computadores.

O presente trabalho tem como objetivo de demonstrar o desenvolvimento de uma aplicação distribuída de Chat, com base na arquitetura Cliente-Servidor, utilizando a linguagem de programação Java.

Sistema Distribuído

Desenvolver uma aplicação distribuída é uma tarefa complexa. Essa complexidade não advém em si da aplicação, mas sim dos processos gerencias necessários para que esta possa ser distribuída. Segundo Tanenbaum [5], um sistema distribuído – SD – é um conjunto de computadores independentes entre si que se apresenta aos usuários como um sistema único. A dificuldade é encontrar maneira de possibilitar que várias máquinas (computadores) comuniquem e troquem informações entre si.

A Fig. 1a e 1b apresentam arquiteturas de SD. Cliente-Servidor Fig. 1(a) e Peer-to-Peer Fig. 1(b) – conhecida também como híbrida.

Utilizou-se da arquitetura Cliente-Servidor para gerar a aplicação, que consiste na distribuição de trabalho entre fornecedores de serviço, e requerentes destes serviços. Os fornecedores são denominados servidores, enquanto os solicitantes chamados de clientes.

A aplicação desenvolvida tem por objetivo a comunicação entre máquinas através de um elemento central – Servidor – que estabelece conexão com todos os outros elementos (nós) – Clientes – permitindo interações entre nós distintos. Considera-se este protótipo como distribuído, uma vez que apresenta aos usuários como uma ferramenta única (transparente), no entanto, é arquitetado possuindo componentes distintos em uma rede.

Elementos fundamentais

Para o desenvolvimento da aplicação utilizou-se da linguagem de programação Java, que segundo ORACLE [3], foi lançada em 1995 pela Sun Microsystems. É uma tecnologia altamente capacitada para atender o desenvolvimento de softwares, como: jogos, aplicações corporativas, dentre outros sistemas. Ainda de acordo ORACLE [3], Java é executada em mais de 850 milhões de computadores pessoais e também em bilhões de dispositivos diversos.

A linguagem é interpretada, e cada dispositivo que tem suporte a linguagem possui uma espécie de interpretador chamado Java Virtual Machine – JVM. Essa JVM lê e executa cada linha assim que é acionada uma instancia do sistema. Este fato proporciona sua portabilidade, que consiste na capacidade de execução em vários ambientes operacionais diferentes, como celulares, desktop, e eletrodomésticos diversos.

A linguagem Java suporta a criação e manipulação de Thread, elemento de fundamental importância para a implementação da aplicação, facilitando o seu uso.

Em diferentes situações se observa a necessidade de executar duas atividades ao mesmo tempo, ou seja, concorrentes. Para isso, faz o uso de Thread`s ou MultiThreading`s. As Threds possuem algumas vantagens em relação aos processos, por exemplo, a troca de mensagens entre threads é mais eficiente que troca entre processos. Os Processos são pequenas porções de instruções/comandos que o sistema computacional fragmenta/subdividi as operações que irão ser executados. A linguagem Java permite criar aplicações multiprocessadas.

Java permite também a utilização de Sockets, outro elemento fundamental para a comunicação entre aplicações em máquinas distintas. De modo geral, consiste em um canal de comunicação entre duas máquinas (computadores) através da abertura de "portas". Por esse canal pode transferir informações de diversos tipos. Para cada Socket aberto, somente uma aplicação pode utilizar o canal da máquina em questão, e é necessário identificação da porta na qual a aplicação irá funcionar. Assim, podemos dizer que Socket, é uma associação única entre IP da máquina e porta da mesma.

Para desenvolver a aplicação fez-se o uso do IDE (Integrated Development Environment) NetBeans, versão 7.4, e para gerar o diagrama de Classe, software Astah Community 6.7.

Resultados

O sistema proposto consiste em uma aplicação de comunicação em rede local (LAN), denominado "Talk: A broadcasting Chat", que permiti pessoas em uma mesma rede trocar mensagens via Broadcast – processo no qual todos os clientes que tiver a aplicação em execução irão receber mensagens.

O diagrama de classe Fig. 2, permite identificar as classes que o sistema possui, sendo elas, Cliente e Servidor, permitindo ainda identificar os métodos de cada uma delas.

Como se pode observar ainda na Fig. 2, há uma multiplicidade em que um cliente só se comunica com um servidor, enquanto o servidor comunica-se com todos os clientes ativos por meio de Broadcast.

Em relação às arquiteturas de SD, mais precisamente a arquitetura Cliente-Servidor, percebe-se que a classe Servidora é o elemento central, pois esta recebe todas as mensagens enviadas pelos clientes ativos, possuindo o papel de processar e redistribuí-la para as demais.

Para o desenvolvimento da aplicação utilizou-se de conceitos e técnicas descritas no item elementos fundamentais, nas quais Socket e Thread foram às bases para o desenvolvimento.

A tabela 1 descreve a lógica utilizada para criação dos métodos necessários para a comunicação entre as classes. Esses métodos estão listados em ordem cronológica na qual a comunicação se estabelece.

A interface principal da aplicação contém os campos necessários para obter a comunicação, sendo eles: **Nick** consistindo no campo destinado ao nome do usuário que está logado - passado no processo de validação de usuário na aplicação; O campo **Diálogo** é o destinado para visualizar as mensagens enviadas pelos Clientes; **Mensagem** é o campo destinado à escrita das mensagens a serem enviadas; **Botão Enviar** chama o método "Enviar", que ao ser clicado, pega o texto no campo de mensagem e envia pelo canal estabelecido via Socket para o servidor.

Existem algumas restrições impostas no método de validação de usuário e validação de mensagens, que foram pensadas a fim de se viabilizar de forma eficaz a comunicação entre os diversos usuários. São elas: não é possível logar um usuário com nome vazio; não é possível enviar uma mensagem vazia. A figura 3 se refere à interface da aplicação desenvolvida.

Considerações finais

O principal objetivo foi fazer efetivamente máquinas distintas trocarem informações através de canais criados via Socket. Este objetivo foi alcançado, porém, compreende-se que um sistema de comunicação eficiente precisa de várias diretrizes que não foram abordados aqui, como por exemplo, segurança, transferência de arquivo (documento, imagens) e etc.

Projetou-se a especificação da aplicação para que contivesse a funcionalidade de envio de arquivos, no entanto, com a complexidade de manipulação e transferência de bytes em Java, impossibilitou a implementação dessa funcionalidade. Mesmo assim, pode-se em trabalhos futuros ajustar a aplicação para corresponder a tal função.

Para trabalhos futuros, sugere-se o desenvolvimento de uma aplicação robusta que vise altos requisitos de desempenho, propondo soluções de melhorias e funcionalidades que o sistema de comunicação necessita para integrar o processo de comunicação, possibilitando um meio único de disseminar a informação para as organizações.

Referências

[1] CESTA, André Augusto. **JAVATUTORIAL - A Linguagem de Programação Java e Orientação a Objetos**. Unicamp - 1996. Disponível em: <http://www.ic.unicamp.br/~cmrubira/JAVATUT14PDF.pdf>. Acesso em 19 de Ago de 2014.

[2] CARVALHO, Fábio Câmara Araújo de,. Gestão do Conhecimento. São Paulo: PEARSON, 2012.

[3] ORACLE. **O que é a tecnologia Java e por que preciso dela?.** s.d. Disponível em:< http://www.java.com/pt_BR/download/faq/whatis_java.xml> Acessado em : 01 de Jun. de 2014.

[4] SOMMERVILLE, Ian. **Engenharia de Software**. - São Paulo : Pearson, Addison Wesley, 2007.

[5] TANENBAUM, Andrew S.; STEEN, Maarten Van., Sistemas Distribuídos: Princípios e Paradigmas. São Paulo: Pearson Pretice Hall, 2007. 2ed.

Tabela 1 - Processo de Comunicação entre Classes. Fonte: Própria – 2014.

Servidor	Cliente
1- Abre a porta através de um ServSocket, especificando a porta – utilizou-se 61000.	2- Cria um Socket, passando assim o IP do Servidor e especificando a porta.
2.1- Aceita solicitação do Cliente e abre o canal de comunicação para envio de mensagens.	3- Criar uma Thread responsável por escutar mensagens enviadas pelo Servidor.
2.1.1- Cria Thread responsável por escutar mensagens enviadas pelos Clientes.	5- Envia mensagem para o Servidor.
4- Se houver mensagem em arquivo, carrega as mesmas para todos os Clientes.	
6- Armazena em arquivo as mensagens enviadas pelos Clientes.	
6.1- Encaminha para todos os Clientes as mensagens enviadas.	

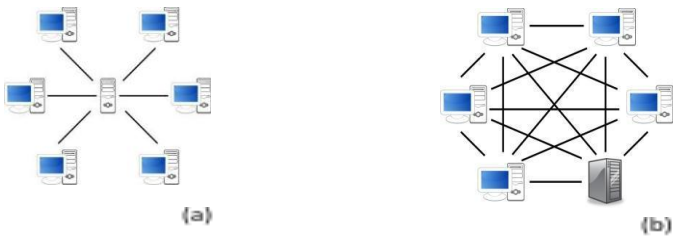

Figura 1. Arquiteturas de Sistema Distribuído. Fonte: WS/tech² s. d.

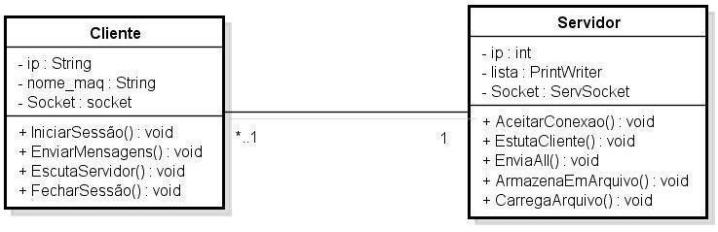

Figura 2. Diagrama de Classe do Sistema Talk. Fonte própria – 2014, utilizando Astah Community 6.7.0.

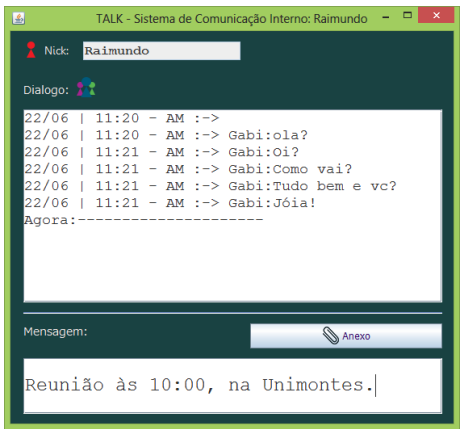

Figura 3. Interface da Aplicação. Fonte própria – 2014, utilizando IDE Netbeans V.7.4.