SERVIÇO NACIONAL DE APRENDIZAGEM COMERCIAL - SENAC
FACULDADE SENAC PORTO ALEGRE
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Yonathan Stein
Calvin Ávila Custódio

RELATÓRIO TÉCNICO-CIENTÍFICO DE SISTEMAS DISTRIBUÍDOS

CANTINA MONETIZADA

Porto Alegre, 2017

SUMÁRIO

1. INTRODUÇÃO 3

2. PROBLEMA 4

3. SOLUÇÃO DA CANTINA MONETIZADA 5

4. CONSIDERAÇÕES FINAIS 6

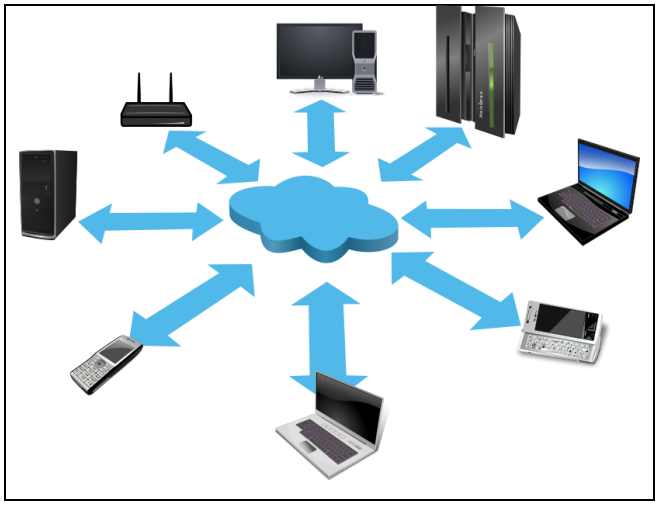
REFERÊNCIAS 7

1. INTRODUÇÃO

Este trabalho tem como objetivo desenvolver uma aplicação chamada Cantina Monetizada utilizando os conceitos de comunicação de Sistemas Distribuídos (SD**)**.

Os Sistemas Distribuídos são um conjunto de máquinas operando em rede, com o objetivo geral de compartilhar recursos e concluir uma tarefa em comum, tornando uma aplicação mais eficiente, logo; possui a capacidade de aumentar o seu desempenho sob carga, particularidade conhecida como **Escalabilidade**. Para que tudo isto seja possível, o SD possui dentre as suas características a possibilidade de execução paralela de tarefas, sendo este o aspecto do **Paralelismo.** Ao mesmo tempo, mediante a **Concorrência**, tolera que estas tarefas sejam executas simultaneamente. É interessante mencionar que um SD é mais do que uma rede com servidores espalhados pela mesma, este envolve certas peculiaridades que não aparecem numa rede comum, característica da **Transparência,** como privar o usuário de qualquer conhecimento sobre como o ambiente executa as suas tarefas (UNESP, 2013). A Figura 1 demonstra um exemplo desta arquitetura.

Figura 1



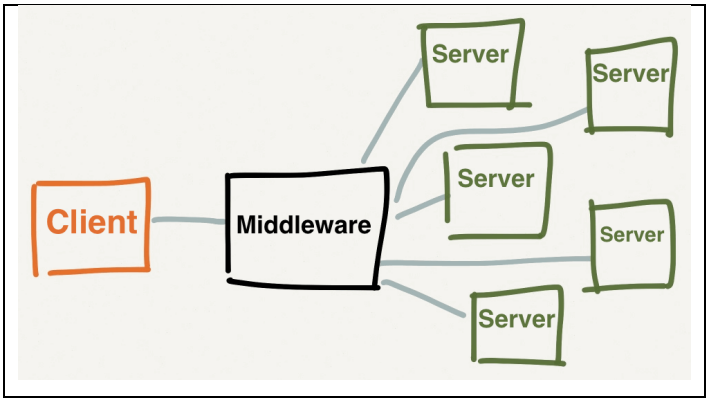
Fonte: Joel Corrêa

Desta forma, com os princípios do SD, o principal foco da aplicação desenvolvida neste relatório é auxiliar as cantinas de faculdades e escolas a realizarem vendas de seus produtos com o modelo pré-pago.

2. PROBLEMA

Um aspecto importante do desenvolvimento de um SD, permeada pela **Heterogeneidade**, é de permitir a execução do código da aplicação em diferentes arquiteturas. Para isto se recorre a um *Middleware*, sendo este uma camada de software que atua como uma ponte entre um sistema operacional ou banco de dados e aplicativos, especialmente em uma rede. A Figura 2 demonstra de forma esquemática este funcionamento (CORRÊA, 2014).

Figura 2



Fonte: Joel Corrêa

Outro ponto fundamental é o **Tratamento de Falhas**, como na tentativa de envio de um pedido e o não recebimento do mesmo pelo servidor, podendo ser repetido o processo um determinado número de vezes sem qualquer nova interação por parte do usuário, havendo uma tolerância, mascarando e recuperando estes dados, e caso não obtenha sucesso será enviado um *feedback* em forma de mensagem para que o usuário possa ter uma tomada de decisão.

Um dos desafios deste sistema tange o aspecto da **Segurança**, sendo necessários manter a integridade dos dados, a disponibilidade do sistema bem como garantir a confidencialidade a partir da autenticação dos usuários.

Pensando em garantir a qualidade deste sistema, mediante a sua natureza **assíncrona**, este irá trabalhar apenas com o protocolo **TCP**, ou seja; em nível confiável de serviço e com garantia de ordem de pacotes (RODRIGUEZ, 2016).

3. SOLUÇÃO DA CANTINA MONETIZADA

Para o desenvolvimento da Cantina Monetizada foi criada uma camada para a autorização de acesso a cada *login,* verificação de saldo e a sua devida consulta junto à base de usuários pré-registrados e seus receptivos valores armazenados. Com isto, as questões referentes a **Segurança**, mesmo que num formato simplificado, foram contempladas.

Foram criados dois projetos a parte, o primeiro representa os possíveis clientes que poderiam se conectar no Servidor e o outro representa o Servidor em si, o qual é responsável por fazer as autenticações dos usuários juntamente com todas as lógicas de negócio envolvidas.

No *backend* foi utilizada uma classe chamada *UsersBuilder,* sendo esta responsável por criar os objetos que representam cada usuário e guardá-los em um *array*. Desta forma, foi possível obter os usuários pré-cadastrados para conectar no sistema. Foi criada outra classe chamada *UsersThread*, a qual é responsável por criar *Threads* representando os clientes que eventualmente iriam se conectar, de forma simultânea, no sistema (CAELUM, 2017).

No *frontend,* no que tange o desenvolvimento ao mesmo tempo em que se visou a simplificação da aplicação, foi criada uma classe responsável por instanciar clientes e para se conectar com o *backend*.

Quanto ao funcionamento do sistema, quando um cliente se conecta com o Servidor será exibido um menu com as seguintes opções de escolha: verificar o saldo atual, realizar pedido, verificar pedidos na fila de atendimento ou sair. A Figura 3 exibe esta etapa de funcionamento do sistema.

Figura 3

```
run:
Connected to Server!
Registration:
1234
Password:
1
Choose one of the options below:
1- Check balance
2- Order dish
3- Show finished orders
4- Exit
```

Fonte: criada pelos autores do projeto

No momento em que a opção de realizar pedido for selecionada, o usuário deverá informar o valor da refeição que deseja. Quando isto ocorre, a lógica de verificação de saldo do cliente em relação ao valor do pedido é ativada. Isto irá determinar se a resposta do servidor será validar a solicitação, emitindo uma confirmação de que o pedido foi adicionado à fila, ou informar que o saldo é insuficiente.

4. CONSIDERAÇÕES FINAIS

Uma dificuldade encontrada se deu na comunicação entre os processos envolvidos, mediante a natureza descentralizada da aplicação, necessitando inclusive de atenção especial para a sua codificação.

A parte referente a protocolos de comunicação, o **TCP**, ocorreu de forma fluída, uma vez em que todo o sistema foi executado dentro do mesmo ambiente ideal e seguro, sem apresentar qualquer interferência externa.

Como a versão inicial do sistema foi puramente desenvolvida em Java *Deskop*, não houve dificuldades quanto a **Heterogeneidade**, porém; para a futura versão e aplicação em caso real este é aspecto fundamental afim de permitir que equipamentos de diferentes fabricantes, versões e sistemas operacionais possam integrar a Cantina Monetizada. Neste caso, será imprescindível o uso de um *Middleware* principalmente pelo fato de diversas cantinas possuírem sistemas adversos.

REFERÊNCIAS

CAELUM. **Programação Concorrente e Threads**. 2017. <https://www.caelum.com.br/apostila-java-orientacao-objetos/programacao-concorrente-e-threads/>. Acessado em: 04/07/2017

CORRÊA, Joel. **Sistemas Distribuídos**. 2014. <http://slides.com/joelcorrea/sistemas-distribuidos#/>. Acessado em: 04/07/2017

RODRIGUEZ, Noemi. Comunicação entre Processos. **PUC-Rio.** 2016. <http://www.inf.puc-rio.br/~noemi/sd-10/aula2.pdf>. Acessado em: 04/07/2017

UNESP. Sistemas Distribuídos. **Departament. de Ciências de Computação e Estatística**. 2013. <https://www.dcce.ibilce.unesp.br/~aleardo/cursos/fsc/cap13.php>. Acessado em: 04/07/2017