

USANDO A LINGUAGEM DE PROGRAMAÇÃO JAVA PARA CRIAR PÁGINAS DINÂMICAS E INTERLIGADAS ATRAVÉS DO JAVA.RMI

*Oswaldo Bassani Neto*¹

*Líria Matsumoto Sato*²

Resumo

Neste trabalho foi estudada a linguagem de programação Java como solução para problemas de cliente-servidor em aplicações em rede. Toda a linguagem foi explorada gerando programas simples que posteriormente tiveram sua complexidade aumentada. Dentre nossos estudos, destacam-se o uso do pacote 'java.rmi' e o uso de banco de dados distribuídos. Quanto ao RMI (Invocações de Métodos Remotos ou 'Remote Method Invocation'), tem-se por objetivo utilizá-lo nas aplicações que requerem 'sockets', usaremos o RMI para uma aplicação cliente-servidor na web, em específico, um jogo de poker com 'chat' (bate-papo) para os jogadores. O uso do banco de dados se dará numa aplicação para pequenos e médios negócios em que duas lojas devem manter a consistência dos seus cadastros de clientes entre outros, para o gerenciamento dos dados, escolheu-se o SGBD mySQL como servidor para desenvolver esta aplicação, o RMI conectará os servidores das lojas sendo o responsável pela consistência e segurança dos dados distribuídos.

Palavras-chave: Java; RMI; invocações de métodos remotos; banco de dados distribuídos; cliente-servidor.

Abstract

In this work it was studied Java programming language as solution for client-server problems in applications for internetworking. The whole language was explored generating simple programs that later had its complexity increased. Among our studies they stand out the use of 'java.rmi' package and the use of distributed database. Regarding RMI (Remote Method Invocation) the objective is use it in applications that most often use sockets. RMI will be used for a client-server web-application, specially a poker game with chatting capability between the players. The database will be applied in an application for small and medium business where two stores should keep data about your clients for example, it was chosen mySQL database server to develop this application, RMI will connect and manage consistency and security of distributed data.

Key words: Java; RMI; remote method invocation; distributed database; client/server.

1. Introdução

Navegando pela Internet notamos que algumas páginas são recarregadas novamente para efetuarem uma simples troca de opção, dado que a taxa média de transferência de dados existente é baixa, achamos importante que novas tecnologias entrem a favor do usuário final livrando-o de 'downloads'

¹ Bolsista do CNPq/PIBIC 2002/2003.

² Professora Doutora do Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP.

desnecessários e aumentando as possibilidades de navegação deste usuário. Atualmente emprega-se para páginas dinâmicas extensões do tipo cgi, asp, html com javascript entre outros. A Internet de banda larga é uma realidade para poucos, a maioria das pessoas ainda está acessando a rede por conexão 'dial-up' (discada) e sabemos que este acesso não tem velocidade muito alta, portanto é importante lembrar que o internauta precisa ficar entretido para não abandonar o 'site' ou serviço que ele está acessando. Vendo esta situação tentamos encontrar uma saída, descobrimos nos 'applets' vantagens e desvantagens. Encontramos vários 'applets' bons e pequenos, facilitando muito o 'download', no entanto, é necessário ao internauta ter o programa que execute o Java. As páginas com 'applets' são muito mais simples e os 'applets' são normalmente armazenados no computador do usuário possibilitando um próximo acesso muito mais rápido. A proposta de fazer páginas pequenas fornecendo ao usuário interatividade não está sendo muito utilizada, pois as páginas atuais que possuem larga capacidade de interação não têm como preocupação o tempo de 'download' e realmente ficam restritas à Internet de alta velocidade. A Java é uma linguagem de programação conhecida por muitos e muito aplicada na Internet, geralmente em 'applets' simples que não se comunicam com outros na rede. Outro uso desta linguagem encontra-se no processamento paralelo que utiliza o RMI para manter comunicação entre processos. Podemos utilizar o RMI para criar sistemas cliente-servidor onde o cliente chama uma função que é executada no servidor, isso restringe o conhecimento do programa pelo cliente o que é um item importante para algumas aplicações.

Visamos difundir a existência das aplicações que usam RMI, tornando esta tecnologia acessível para os programadores da Internet, mostrando como criar programas e páginas que deixem o usuário interagir e cuja resposta seja rápida. Queremos também criar um programa para uma aplicação real que demonstre as possibilidades do uso do RMI para aplicações comerciais. Nosso objetivo é tornar possível que o banco de dados deixe de ser centralizado e passe a ser distribuído. Para isso precisamos de programas que tornem essa distribuição transparente para o usuário.

Escolhemos para o nosso banco de dados um servidor pequeno e barato dentre os disponíveis a nós pela Internet, o MySQL. Nas aplicações para pequenas e médias bases de dados encontramos muitas vezes um banco de dados em Access, tratado pelos usuários como grandes tabelas. Na maioria dos casos poderíamos usar o MySQL, pequeno e gratuito servidor de banco de dados. A falta de soluções para pequenas e médias bases de dados torna-se um limitante de lucros em muitos casos, dado que os usuários do sistema têm muito trabalho para se locomoverem pela base de dados.

Para usar o Java com o MySQL criamos uma coleção de classes para o tratamento desta comunicação, usando como base o sistema conhecido como JDBC, neste caso o mysql-connector-java-2.0.14. Visamos com estas classes proteger o sistema de problemas de comunicação entre os nós e conseqüentemente de inconsistências geradas por estas falhas.

A motivação do uso da linguagem de programação Java decorre do fato dos sistemas implementados em Java estarem em destaque, não pelo seu alto desempenho, mas por sua portabilidade, ou seja, são bons candidatos para projetos de pequenos 'grids' heterogêneos. A Java vem se destacando também pelo seu uso na Internet, demonstrando que mais pessoas possuem algum conhecimento

na linguagem, isso facilita o desenvolvimento de aplicações dentro de empresas, projetos ou uso geral.

2. Materiais e Métodos

O material disponível para pesquisa é muito vasto, a Internet é uma fonte de programas e informações muito grande e mereceu atenção. Livros mais específicos enfocam determinado assunto de forma mais ampla e com um conteúdo mais técnico. O princípio básico para criação de novos programas em Java foi a procura de aplicações parecidas aquelas que queríamos criar, as tecnologias existentes (cgi, asp,...) nos mostram o que devemos implementar. Analisando estas implementações podemos acrescentar ou retirar itens para que a aplicação sirva a nossa necessidade e interesse.

No decorrer do projeto foram usadas varias plataformas para o desenvolvimento dos programas, entre elas destacam-se Linux (Red Hat e KDE) e Windows (98, 200, XP). Este fato serviu a nosso interesse sendo um grande teste da portabilidade da linguagem. Também foram usados vários computadores, de Pentium I até III e um AMD Athlon XP 2000.

Nosso método para confecção dos programas é um pouco diferente dos métodos convencionais. Encontramos na Internet vários exemplos de aplicativos em Java, para o estudo da linguagem tomamos como base cada um dos programas escolhidos e por um processo de análise do código aprendemos muito da linguagem. Partindo de um programa base, como por exemplo um demo da Sun, podemos isolar blocos funcionais desejados para tornar possível a extração do que nos interessa.

É interessante expor aqui que esta é uma forma de estudar e conhecer a linguagem Java por completa, encontrar um ponto de interesse no meio do programa exemplo requer atenção do programador, que deve entender tudo que esta acontecendo no programa e assim ampliar sua visão sobre a linguagem. Conversando com alunos e professores percebemos a existência de modos diferentes de estudo. Esta forma que estamos mostrando apesar de simples não é comum, muitos preferem obter o conhecimento em documentos de ajuda mais detalhados. Esses documentos são muito interessantes para os usuários já iniciados na programação, no entanto acreditamos que para uma iniciação na linguagem é muito mais vantajoso um aprendizado por exploração de exemplos.

Para expor nosso método e parte do trabalho desenvolvido mostraremos os programas por nos desenvolvidos durante este projeto, partindo dos primeiros passos da linguagem em aplicações comuns até as aplicações inovadoras de banco de dados distribuídos com um aplicativo cliente-servidor usando RMI.

3. Resultados

3.1. Aplicativos simples

Como queremos construir um acervo de exemplos armazenamos todos os programas encontrados na rede, alguns programas foram estudados e aproveitados,

outros serviram de inspiração para uma nova confecção. Muitos programas exemplos que encontramos eram grandes e por isso não servem para a iniciação do programador nesta linguagem. Logo, os programas que criamos para fim introdutório são menores e possuem poucas funcionalidades. Desta forma, com a atenção direcionada para um ponto do programa, tornamos nosso exemplo mais didático.

Mostraremos abaixo alguns exemplos que confeccionamos.

Com a preocupação de introduzir novos usuários à programação Java, tomamos o cuidado de não cometer os erros mais comuns dos exemplos que encontramos com esta finalidade. Geralmente os sites que visam o aprendizado da linguagem esquecem que o programador precisa instalar o Java e as ferramentas para editá-lo. Nosso primeiro exemplo é um teste da instalação do Java do usuário.

Em uma página 'html' mostramos como o usuário deve proceder para instalar o Java em sua máquina e trazemos um 'applet' simples para introduzi-lo a linguagem.

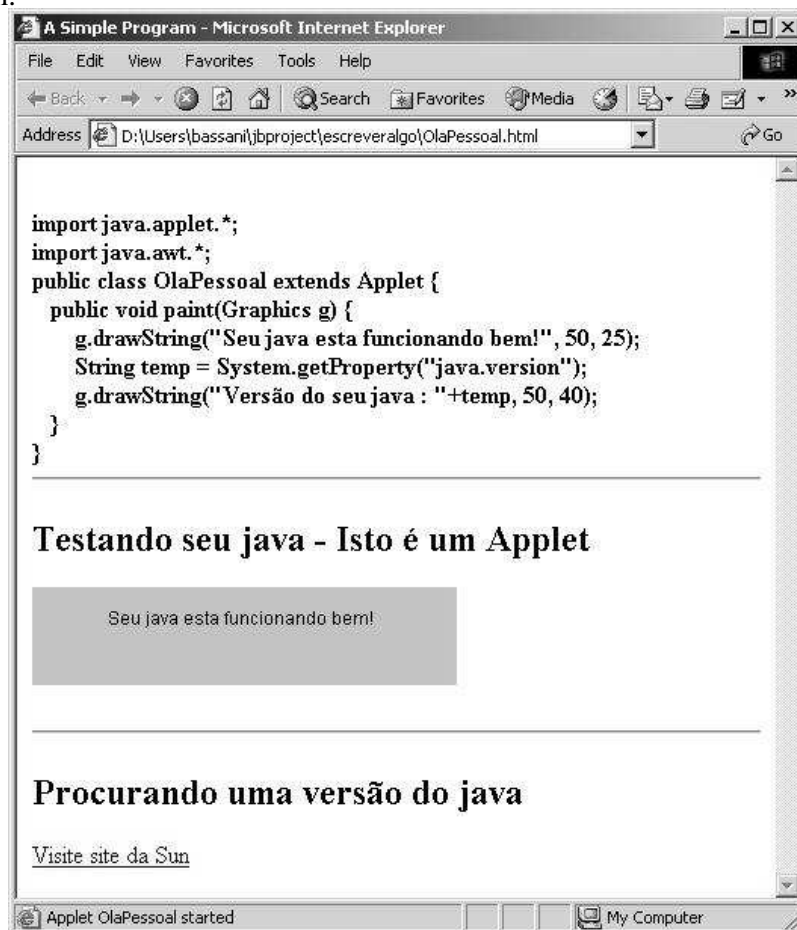


Figura 1. Primeiro 'applet'

Partindo do mesmo exemplo podemos adicionar mais funções ao nosso exemplo.

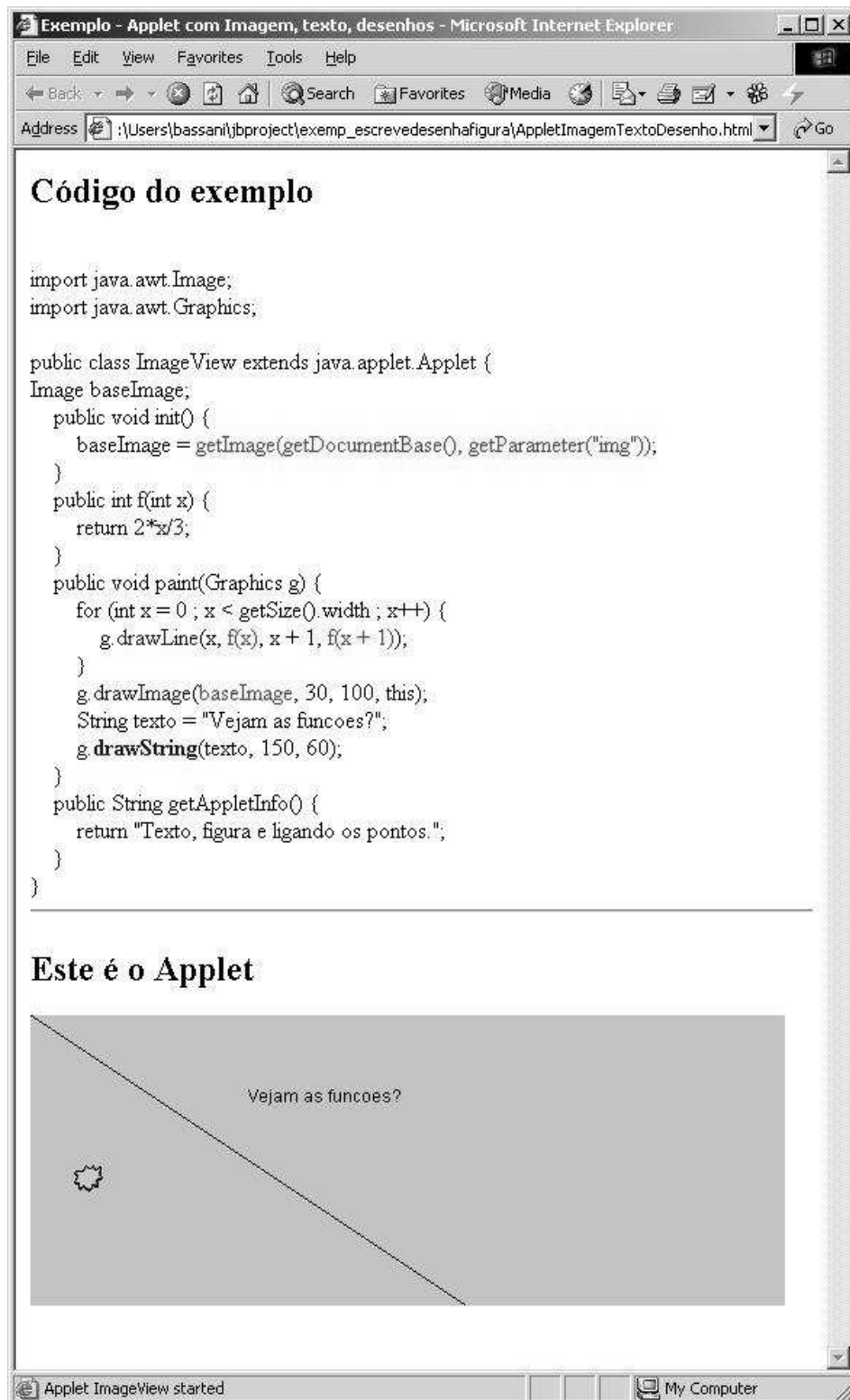


Figura 2. Exemplo de ‘applet’ simples com mais recursos

Um exemplo bem interessante foi um ‘applet’ que desenhava funções matemáticas e era controlado por uma interface na página ‘html’ que o continha.

Para usar nosso exemplo anterior para desenhar gráficos basta trocar o código da função $f(\text{int } x)$. Vamos desenhar agora um co-seno, para isso a função passa a ser:

```
double f(double x) {
    return (Math.cos(x/6) + 2) * getSize().height / 4;
}
```

Alterando a frequência da forma de onda podemos observar se estamos fazendo uma boa amostragem ou se precisamos mais interações para desenhar a curva com perfeição.

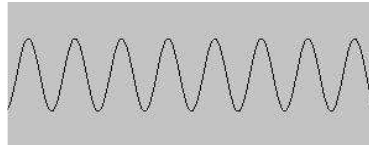


Figura 3.1. Um co-seno frequência menor

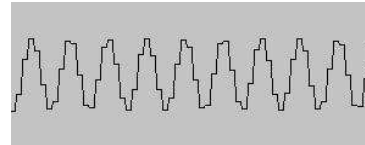


Figura 3.2. Maior frequência com mesma taxa de amostragem para o desenho

Para demonstrar como é possível criar uma boa interação podemos mostrar este exemplo destinado à apresentação de gráficos de equações, através de uma interface em 'html' o usuário seleciona a equação e os parâmetros desejados, os botões chamam métodos do 'applet' fazendo com que esse desenhe a curva pedida.

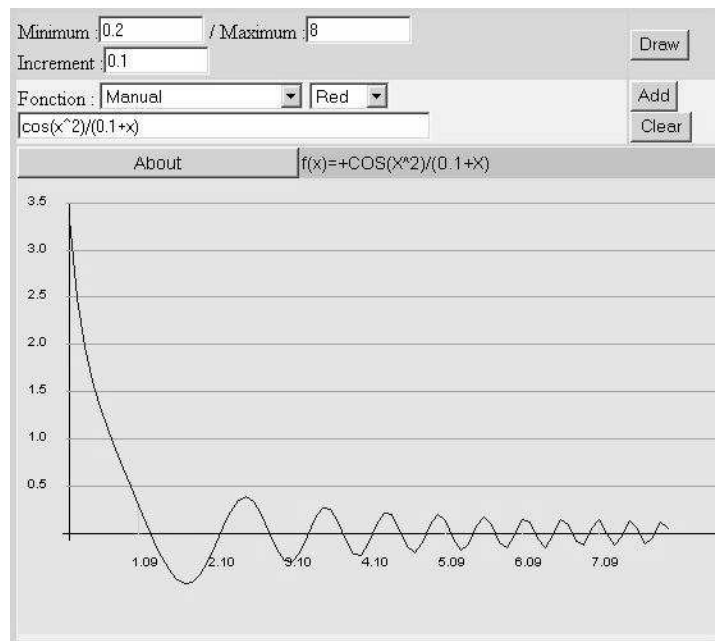


Figura 4. Interface 'html' de um aplicativo chamado "tFunctionV1.0" para gráficos

Para acessar as funções de um 'applet' podemos criar um 'html' conforme o indicado abaixo:

- Devemos ao acrescentar o 'applet' nomeá-lo, como mostrado abaixo.

```
<applet code=aplicativo.class name="nome_applet">
```

- A classe aplicativo.class deve implementar a função `criaLinks()`.

- Para executar a função podemos colocar um botão no 'html' que a chame:
`html_resposta = nome_applet.criaLinks();`

Esse programa retorna uma sequência de 'links' que podemos introduzir agora na página através de javascript sem a necessidade de recarregar a página, obtendo a página abaixo.

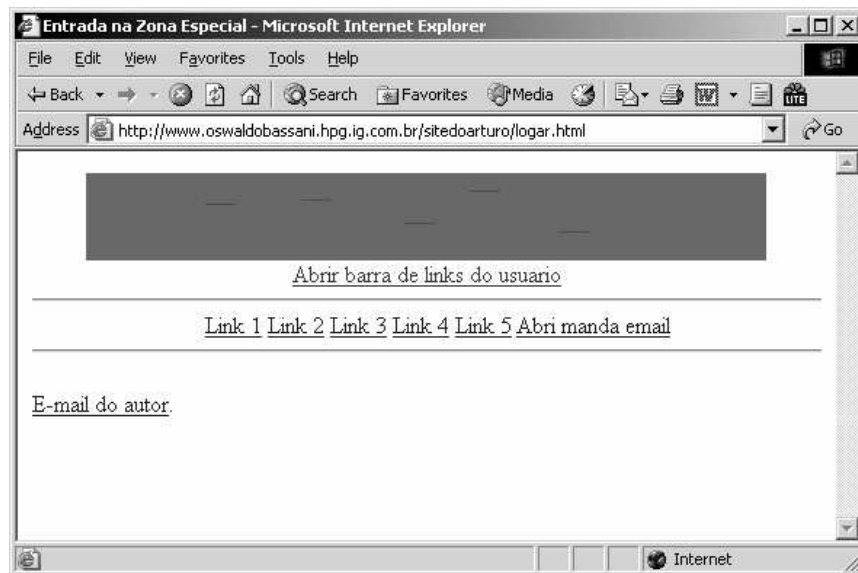


Figura 5. Página com 'links' gerados pelo 'applet'

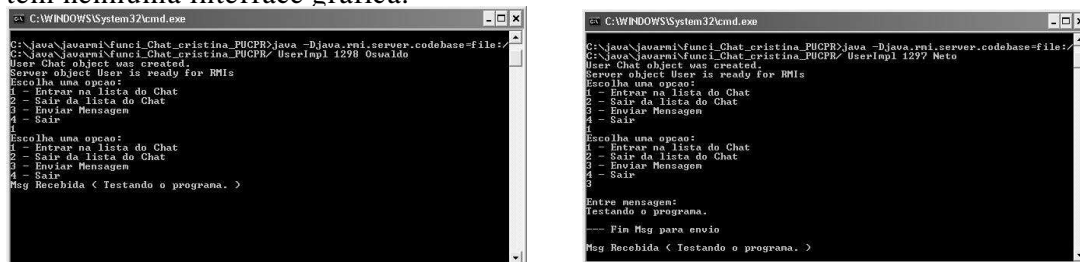
Podemos também incluir aqui programas que não tenham nenhuma forma de interface, mas que podem ajudar o usuário no seu dia-a-dia criando programas que sirvam para facilitar certas atividades. Neste exemplo, mostramos um código para copiar um arquivo.

```
import java.io.*;
public class Copy {
    public static void main(String[] args) throws IOException {
        File inputFile = new File("ArquivoOrigem.txt");
        File outputFile = new File("Arquivo Destino.txt");
        FileReader in = new FileReader(inputFile);
        FileWriter out = new FileWriter(outputFile);
        int c;
        while ((c = in.read()) != -1) out.write(c);
        in.close();
        out.close();
    }
}
```

No exemplo acima temos apenas uma base, a utilidade e eventuais aplicações podem ser encontradas pelo programador, se adequando a necessidade e as novas idéias do mesmo.

Abordaremos agora o pacote RMI, tópico importante do nosso projeto. Iniciando com a pesquisa em discussões eletrônicas que se baseavam em programas exemplos disponíveis na internet pudemos construir um programa muito simples de ‘chat’. Construímos primeiramente os arquivos responsáveis pela comunicação, desta forma poderemos expandir o aplicativo acrescentando posteriormente uma interface gráfica.

Baseamos nosso estudo no programa de ‘chat’ disponível no site da professora Cristina V. Pérez Barrios de Souza da PUCPR (<http://www.ppgia.pucpr.br/~cristina/>). Usando a mesma técnica descrita nos primeiros exemplos podemos extrair deste exemplo as funções básicas que iremos precisar, reformulando-as para servirem ao nosso propósito e consertando alguns defeitos encontrados, um deles no descadastramento de usuários. Este exemplo não tem nenhuma interface gráfica.



```

C:\WINDOWS\System32\cmd.exe
C:\java\javaw\func1_Chat_cristina_PUCPR>java -Djava.rmi.server.codebase=file:/
C:\java\javaw\func1_Chat_cristina_PUCPR>User Chat object was created.
Server object User is ready for RMI:
Escolha uma opção:
1 - Entrar na lista do Chat
2 - Sair da lista do Chat
3 - Enviar Mensagem
4 - Sair
1
Escolha uma opção:
1 - Entrar na lista do Chat
2 - Sair da lista do Chat
3 - Enviar Mensagem
4 - Sair
3
Msg Recebida < Testando o programa. >

C:\java\javaw\func1_Chat_cristina_PUCPR>java -Djava.rmi.server.codebase=file:/
C:\java\javaw\func1_Chat_cristina_PUCPR>User Chat object was created.
Server object User is ready for RMI:
Escolha uma opção:
1 - Entrar na lista do Chat
2 - Sair da lista do Chat
3 - Enviar Mensagem
4 - Sair
1
Escolha uma opção:
1 - Entrar na lista do Chat
2 - Sair da lista do Chat
3 - Enviar Mensagem
4 - Sair
3
Entre mensagem:
Testando o programa.
--- Fin Msg para envio
Msg Recebida < Testando o programa. >

```

Figura 6. Chat com RMI – usuário #1 e #2

Voltaremos a esse exemplo posteriormente para inclusão de uma interface.

Para continuar com alguns exemplos passaremos agora para programas não ‘applets’, são aqueles que usam os componentes “Swing” do Java. Eles também podem ser usados em páginas ‘html’ junto com os applets, basta que o ‘applet’ tenha um botão ou ative automaticamente este outro componente.

3.2. Aplicativos com interface

Para iniciar o uso de interfaces o exemplo abaixo traz em particular como trabalhar com figuras e o uso de botões, além disso, temos a familiarização com a classe URL como ponto principal. Este é um exemplo simples que não traz nenhuma complicação.

```

procura = new JButton("Procurar");
//Definindo a ação do botão “Procurar”
procura.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //Pegando o endereço do campo “Endereço”
        ending = end.getText();
        try {
            //Criando o objeto URL
            URL urlimg = new URL(ending);
            //Criando a imagem a partir do URL
            img = new ImageIcon(urlimg);
        }
        catch(Exception exception){}
        status.setIcon(blank);
    }
});

```



```

//Testando o status do processo de exibição da imagem
if(img.getImageLoadStatus()!=8)
    status.setText("Nao foi possivel obter o arquivo.");
if(img.getImageLoadStatus()==8) {
    if(exibestate) status.setIcon(img);
    status.setText("Download completo!");
    temfig=true;
}
frame.pack();
});

```

O objetivo principal com exemplos tão simples é não assustar o programador e deixá-lo com curiosidade de coisas mais complicadas.



Figura 7. Interface simples e pouco complexa

Uma aplicação um pouco mais complexa envolve o “Look & Feel” do Java, esta implementação faz com que os aplicativos ‘Swing’ possam mudar de aparência dado à vontade do programador, ou se for implementado, do usuário. Utilizando um primeiro exemplo destinado simplesmente a atingir o aspecto de “Look & Feel” do Java iremos acrescentar algumas outras funcionalidades como um ‘Label’ (etiqueta) para exibir o número de cliques no botão, ‘Label’ com figura e cor de fundo e um grupo para entrada de usuário e senha.

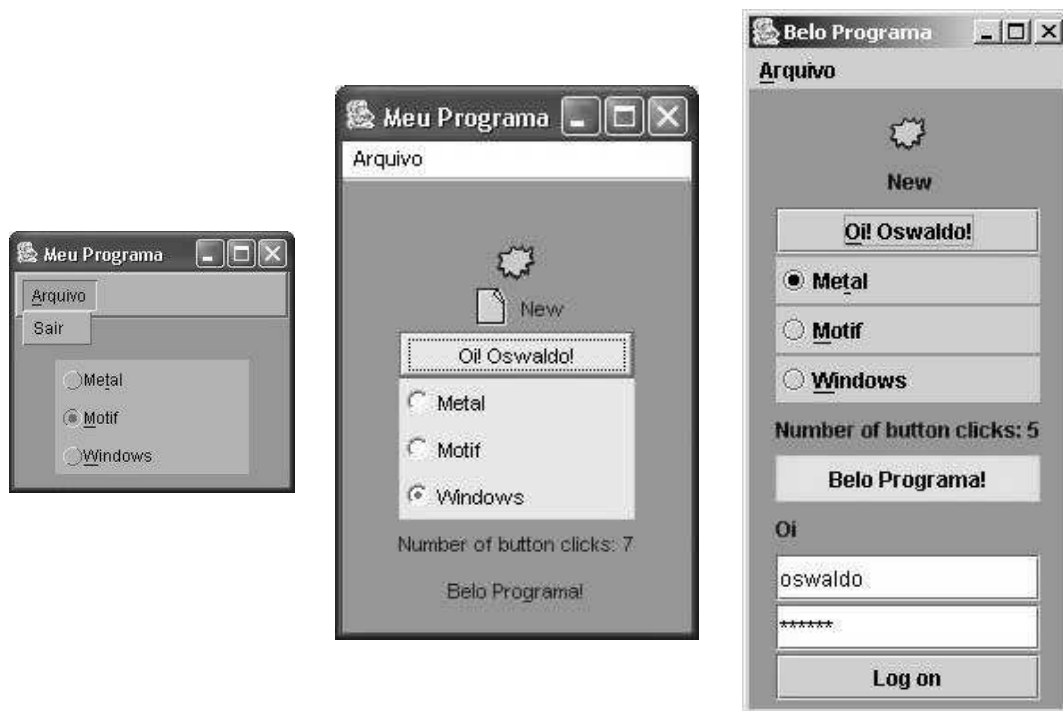


Figura 8. Exemplos de como trabalhar com “Look & Feel”

Para completar o programa de ‘chat’ por RMI desenvolvemos uma interface para o uso deste programa de forma que o usuário final tenha maior facilidade para usar o aplicativo.

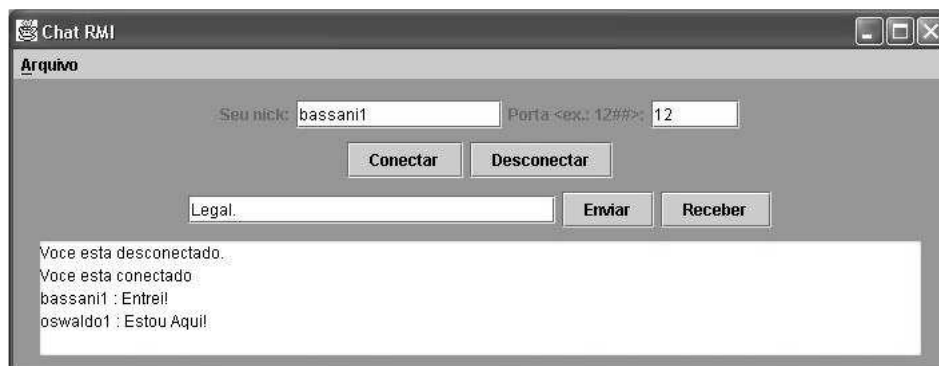


Figura 9. Interface para o programa de ‘chat’

Outro programa que foi desenvolvido foi um jogo de ‘poker’ para um único jogador. Usando um princípio de pontos por combinação de cartas o usuário acumula pontos a cada jogada considerada boa que ele consegue.



Figura 10. Jogo de 'poker' para 1 jogador

Juntando estes dois programas criamos uma terceira variação muito mais interessante. Passamos a ter agora um programa que possui um jogo de 'poker' individual ou em rede, neste ultimo o jogador pode usar o serviço de 'chat' para se comunicar com os outros jogadores.

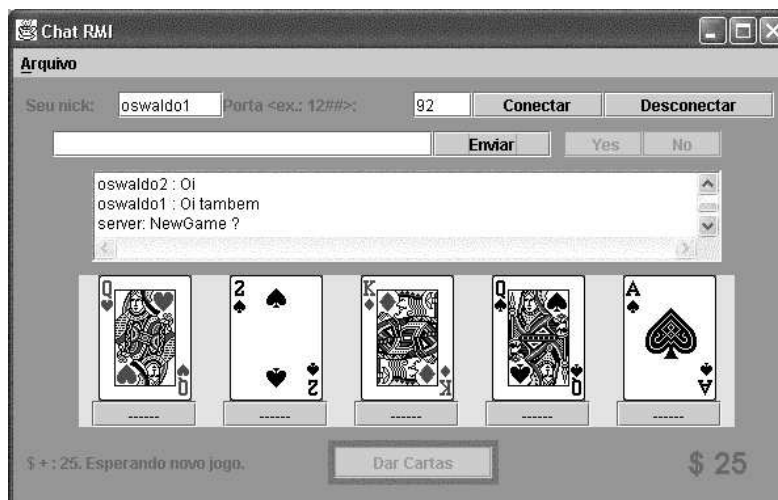


Figura 11. Jogo de 'poker' em rede com 'chat'

Prosseguindo com o estudo da Java iremos agora estudar o relacionamento da Java com banco de dados para posteriormente incluir o RMI para controle das 'queries' enviadas para o servidor de dados.

Dada a importância dos bancos de dados e tendo em vista nosso objetivo achamos indispensável direcionar nossa pesquisa para a comunicação da Java com banco de dados. Como já dissemos estaremos usando o MySQL e para isso vamos usar também o mysql-connector-java-2.0.14 pacote adicional do Java para a comunicação do Java com o MySQL. Criamos uma interface para o usuário criar suas 'queries' e executá-las, não foi objetivo aqui criar um ambiente que crie as 'queries' automaticamente, mas sim um ambiente que facilite o aprendizado das 'queries' pelo usuário.

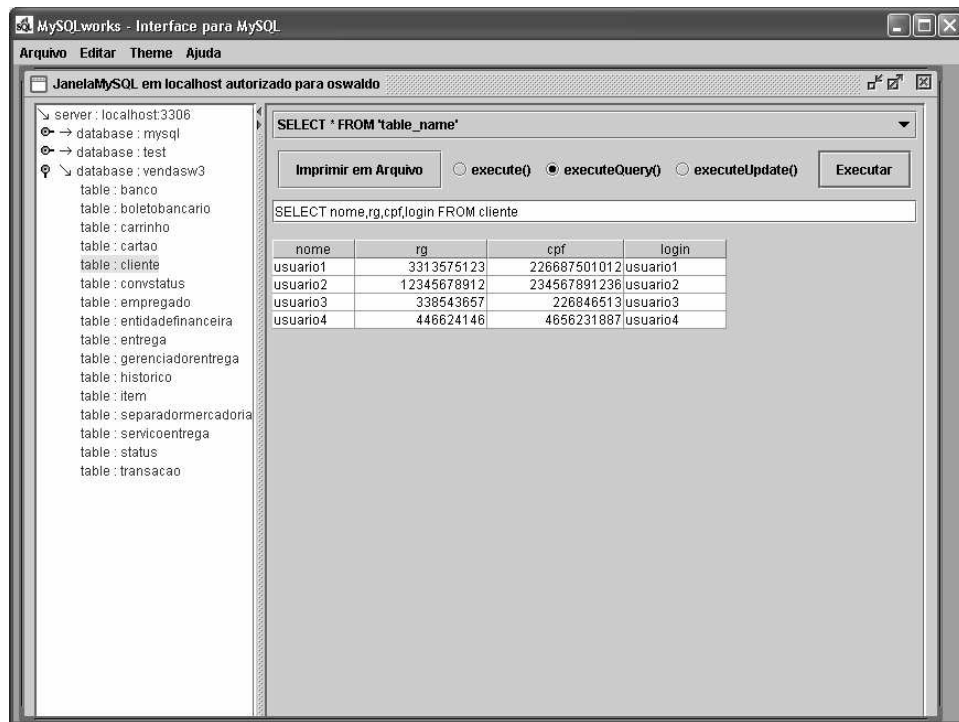


Figura 12. Interface do executor de ‘queries’

Este programa é o início do trabalho com banco de dados e sua função é ser a nossa ferramenta para os próximos passos.

3.2. Aplicativo para solução de problemas reais

Este aplicativo foi o objetivo principal do nosso projeto, é uma solução para banco de dados distribuídos que usa o RMI para manter os dados consistentes. Para concretizar nossa solução definimos um problema e implementamos nossa solução para resolvê-lo.

Nossa escolha foi propor este problema no caso de uma locadora de vídeos. A locadora de vídeos seria composta por duas lojas. Elas se situariam numa mesma cidade, a distância entre elas não seria muito grande tornando possível a troca de filmes entre elas através de um serviço de moto que levaria cerca de vinte minutos para chegar de uma loja para a outra. Um cliente cadastrado seria reconhecido nas duas lojas podendo efetuar locações, pagamentos e devoluções em ambas.

A forma mais comum de implementar uma solução para este problema é o uso de um servidor central de dados que pode ficar em uma das lojas enquanto que a outra deve manter uma conexão permanente para acessar e manipular as informações. Esta solução tem como vantagem a inexistência de preocupações com múltiplas bases de dados, no entanto traz como desvantagem a necessidade de um servidor grande e uma conexão permanente precisando que ambos sejam confiáveis. Uma outra forma é o uso de um pequeno servidor para cada loja, nesse caso a consistência de informações obriga a existência de uma conexão permanente para as eventuais mudanças nos dados.

A solução que propomos torna possível que a conexão entre as lojas possa ser interrompida sem que as lojas parem de funcionar corretamente e caso um dos servidores fique indisponível isso não torna a outra loja indisponível. Cada loja

possui seu pequeno servidor que utiliza para guardar suas informações, a informação dos clientes e parte das informações das fitas de vídeo da outra loja.

Através de um programa de acesso que direciona os acessos ao banco de dados é possível separar os comandos de pesquisa dos comandos de alterações. Comandos de pesquisa são passados diretamente para o servidor local de dados. Já os comandos que alteram os dados são direcionados para um servidor que monitora as alterações visando armazenar comandos importantes para transmiti-los à outra loja, este servidor se encarrega de alterar os dados de maneira segura e mantém a coerência de informações entre os servidores de dados.

A figura mostra como a nossa solução se baseia na arquitetura cliente-servidor e como o RMI está sendo usado. Além da comunicação entre clientes e servidor, a comunicação entre servidores também é feita pelo RMI. Os acessos diretos ao banco de dados só ocorrem no caso de comandos do tipo “SELECT”.

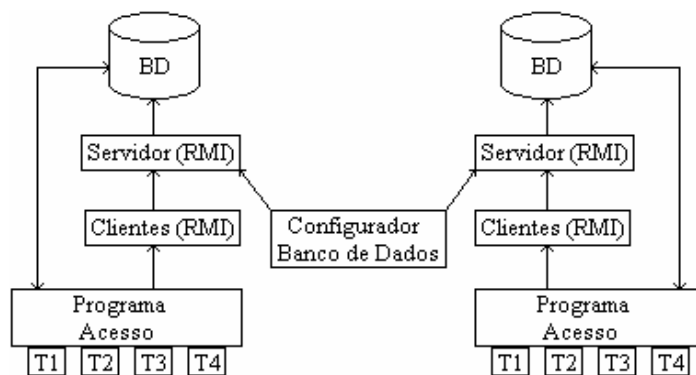


Figura 13. Configuração do Cliente-Servidor-BD da solução

Para testar a solução precisamos injetar ‘queries’ em nossos bancos de dados, para isso criamos uma interface para efetuarmos testes. A nossa plataforma de teste fornece o apoio necessário para a execução dos testes sobre os códigos dos servidores que interligam as lojas e o código dos clientes que se comunicam com estes servidores, podemos então testar a implementação de nossa solução. Também é possível usar um acesso direto pelo Java ao MySQL, retirando do teste o nosso modelo de solução.

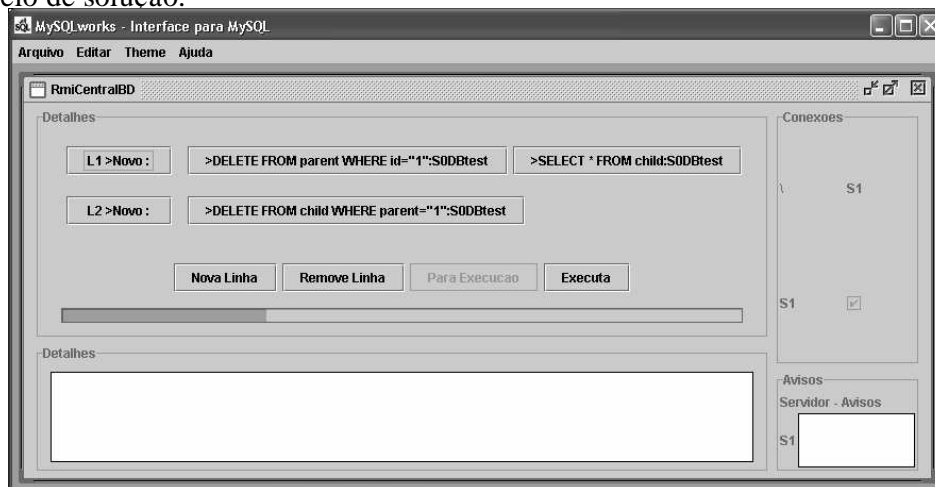


Figura 14. Interface da plataforma de teste

O pacote de comunicação na nossa solução se baseia em quatro classes que valem ser detalhadas. Para esta implementação precisamos da classe de interface do servidor, do servidor, do cliente e de uma classe para conectar o java com o MySQL, de forma semelhante com o JDBC original do mysql-connector-java.

O arquivo de interface, `interfaceRMIDBCserver.java`, possui uma implementação no arquivo `RMIDBCserver.java`, o servidor. Abaixo temos o arquivo de interface.

```
package mySQLworks.geral;
import java.rmi.* ;
import java.sql.*;
import javax.swing.JTable;
import java.util.Vector;
public interface interfaceRMIDBCserver extends Remote {
    public int criaConexao(String user, String pass) throws RemoteException;
    public void destroiConexao(int id) throws RemoteException;
    public void execute(int id, String query) throws RemoteException;
    public void executeQuery(int id, String query) throws RemoteException;
    public void executeUpdate(int id, String query) throws RemoteException;
    public Vector getMessage(int id) throws RemoteException;
    public void clearMessage(int id) throws RemoteException;
    public Vector getMessage() throws RemoteException;
}
```

Abaixo vemos parte da implementação do servidor, mais especificamente o código da função `main`.

```
public static void main (String args []) {
    RMIDBCserver server ;
    if(args.length!=2){
        System.exit(-1);
    }try{
        System.setProperty("java.rmi.server.codebase","file:C:- Local do
Server do cliente");
        Registry.createRegistry(Integer.parseInt(args[1]));
        server = new RMIDBCserver();
        Naming.rebind("rmi://" + args[0] + "/RMIDBCServer", server);
    }catch (Exception e) {
        e.printStackTrace() ;
    }
}
```

Obviamente é necessário conhecer todo o código para compreender o funcionamento por completo desta implementação, citamos aqui apenas a forma como a implementação foi dividida e como se escreve a função principal de um servidor RMI. Para esclarecer o usuário iniciante na linguagem é necessário explicar exatamente como proceder para criar um servidor RMI. A Java obriga que antes de um servidor RMI ser iniciado é necessário estar com o componente “`rmiregistry.exe`” em execução e não devemos esquecer que a compilação deve ser finalizada com o uso do “`rmic.exe`” para as classes que implementarem as interfaces que estenderem o “`Remote`”.

4. Discussão

A linguagem Java é muito versátil, ela pode ser usada em diversos formatos: “applets”, ‘janelas’ e existem também as páginas ‘jsp’ que também entram nesta diversidade de tipos. Além disso, ela foi concebida para ser suportada por qualquer sistema operacional. Através do “Java Virtual Machine” é possível rodar um aplicativo Java em qualquer um dos sistemas disponíveis atualmente.

Uma das questões levantadas no projeto é a velocidade dos aplicativos para o usuário na rede. A velocidade do aplicativo é composta por três períodos durante o processo de utilização. A primeira etapa é o ‘download’ do aplicativo para o computador do usuário, seguido do desempenho do programa às entradas do usuário que não precisam passar por consulta em um servidor na Internet, e por último a velocidade de comunicação do aplicativo com o servidor, caso seja necessário. O programador pode interferir nestas etapas criando aplicativos pequenos e que não necessitem de muito processamento além de disponibilizar servidores rápidos e com banda suficiente para atender a demanda. O usuário fica limitado pelo seu acesso a Internet e a capacidade de processamento do computador.

A implementação por RMI mostrou-se satisfatória embora haja algumas limitações com o uso desse pacote. Um dos problemas com que nos deparamos foi a necessidade de se usar parâmetros serializáveis. Na implementação usando RMI notamos que é preciso definir tudo que se deseja passar de um lado para o outro através do RMI, e após isso, deve-se verificar se tudo é passível de serialização, pois caso haja algum item que não possa ser serializado não será possível usá-lo.

Durante nossa pesquisa notamos que o RMI é pouco usado em aplicativos para internet, seu uso fica muito limitado a pesquisas. A maioria dos exemplos encontrados possuía erros ou informações incorretas de uso. Para se usar corretamente o RMI foi necessário muito estudo. Poucas eram as fontes que explicam de forma clara e correta o passo a passo do uso do RMI, da compilação à execução.

Alguns parâmetros necessários no RMI são chatos de configurar, principalmente a propriedade da pasta do servidor, este parâmetro indica aonde se encontram os arquivos responsáveis pelo servidor RMI. Dependendo do sistema operacional usado a forma com que deve ser escrito este endereço muda e o aplicativo pode não mais funcionar. A escolha da porta de registro pode ser um outro ponto problemático, caso haja mais de um aplicativo que utilize aquele valor de registro, ocorrerá, dependendo do código usado para iniciar o servidor, um erro com o qual o servidor não poderá ser iniciado.

Em relação às desvantagens do Java devemos considerar que além de ser um pouco lenta, os programas em Java permanecem com seu código desprotegido, não sendo possível criar um arquivo ‘exe’ a partir das classes, e as soluções para este problema são caras. Dessa forma é necessário rever o uso da Java para aplicações comerciais, se ela for realmente usada será necessário separar o código (programa) em dois, um cliente (disponível em qualquer máquina) e um servidor (em uma máquina protegida), é uma forma primitiva de proteção, no entanto fornece a segurança necessária. Aparecem como solução alguns ‘ofuscadores’ gratuitos e ou

de baixo custo que embaralham as classes do programa gerado dificultando que o código fonte do programa seja acessado.

Quanto aos bancos de dados, o uso de pequenos bancos de dados distribuídos é uma ótima forma de armazenar dados, no entanto requer inúmeros cuidados para que não apareçam incoerências pelo caminho. Criar soluções padrões nessa linha aponta como algo muito promissor dado a necessidade de se obter maior descentralização para evitar os problemas de comunicação com as redes, além de possibilitar o uso de servidores gratuitos de banco de dados.

5. Conclusões

A linguagem Java aponta como sendo muito promissora em aplicações na internet, é uma linguagem que dependendo do que se usa dela apresenta-se como uma linguagem fácil que pode ser estudada por qualquer aspirante a programador sem a menor dificuldade. Seu uso na internet demonstra que os “programadores web” a utilizam sem problemas.

Aplicativos mais complexos requerem maior estudo em linguagem de programação e necessitam maior atenção e dedicação do programador, tanto para uso como confecção.

A comunicação pelo RMI é uma opção que apesar de necessitar certo estudo mostrou-se interessante, seu não uso em aplicativos na rede se deve a falta de experiência dos programadores nessa linha e pelos exemplos não funcionais que se encontram disponíveis na rede. Acreditamos que se exemplos corretamente descritos e com explicações detalhadas de como executá-los corretamente estivessem disponíveis seríamos capazes de disseminar o uso desta tecnologia de comunicação pela Internet.

O uso da Java em aplicativos comerciais requer muitos cuidados e aparentemente não é uma das melhores escolhas. A necessidade de se ter o pacote do Java instalado no computador do usuário torna necessário na instalação do software uma verificação e a instalação do Java para que seja possível executar o programa, ou então agregar no software todo o conjunto necessário para executar corretamente o aplicativo.

Existe também a preocupação com a proteção do código que requer o uso de ‘softwares’ caros para criação de executáveis que protejam o código ou programas que ofusquem o código tornando-o ilegível e protegendo-o assim da engenharia reversa. Entra aqui o tamanho da empresa responsável pelo projeto, no caso de instituições de ensino é difícil ter o interesse de proteger o código, no entanto, se isso for desejado será difícil de implementar essa proteção já que as soluções disponíveis possuem preços altos. No caso de uma empresa de software é possível comprar uma solução para os problemas aqui citados.

Quanto aos sistemas para banco de dados disponíveis no mercado notamos a falta de aplicações que visam pequenas e médias bases de dados. Logo, pesquisar soluções para este nicho comercial é algo promissor, possibilitando até uma expectativa comercial para solução criada. Ficamos assim satisfeitos com nosso estudo nessa área e com os bons resultados que obtivemos com a implementação que confeccionamos.

Podemos concluir que nosso trabalho foi bem proveitoso, atingindo todas as nossas expectativas com sucesso. O uso da Java para criação de páginas dinâmicas ou aplicativos de entretenimento ou serviço é muito promissor, esperamos que com um incentivo inicial através de sites tutoriais essa linguagem possa ser difundida pela Internet, agregando novos programadores à grande equipe de desenvolvedores da 'web'.

Esperamos que a descentralização de dados e os aplicativos na configuração cliente-servidor sejam cada vez mais frequentes, tornando o RMI e os bancos de dados distribuídos ferramentas interessantes para as novas soluções.

Agradecimentos

Ao CNPq pela concessão da Bolsa de Iniciação Científica.

Referências Bibliográficas

1. FRAIZER, COLIN; BOND, JILL; API Java Manual de Referência Pacotes de API java.applet e java.awt, Tradução Álvaro Rodrigues Antunes, Makron Books.
2. ORTALI, ROBERT; HARKEY, DAN; Client/Server Programming with Java and Corba, Wiley Computer Publishing
3. GOSLING, JAMES; YELLIN, FRANK; The Java Application Programming Interface, Vol1 Core Packages, Addison-Wesley Publishing Company
4. GOSLING, JAMES; YELLIN, FRANK; The Java Application Programming Interface, Vol2 Window Toolkit and Applets, Addison-Wesley Publishing Company
5. Site oficial da Sun : <http://java.sun.com/>
6. Site oficial do MySQL : <http://www.mysql.com/>