Projeto Caixa Eletrônico

Neste documento há a especificação de um protocolo da camada de aplicação com arquitetura Cliente-Servidor para simular funcionalidades de um caixa eletrônico em uma implementação multi-thread em que podem ser abertas várias guias de terminal representando os clientes, desde que o servidor seja executado primeiro para que o cliente possa utilizar as funcionalidades disponíveis. Cada requisição do cliente com o servidor é criada uma thread para tratar tal conexão e logo depois da troca de dados a conexão entre os dois é encerrada.

Servidor

As funções básicas do servidor são as de aceitar conexões de múltiplos clientes, aceitar os comandos e armazenar e manipular os dados enviados pelos clientes. As funções são possibilitadas após a leitura de um arquivo chamado **"contas.txt"** que está na pasta do servidor, no qual está presente todas as informações das contas abertas. São três informações para cada conta: nome, senha e saldo, e no arquivo cada informação é separada por uma vírgula, e a cada quebra de linha equivale a uma conta aberta. Além das classes *ServerSocket* e *Socket* que cuidam, respectivamente, do Servidor e do Cliente.

Cliente

A função básica do cliente é interagir com as funcionalidades presentes no terminal. Navegar entre menus, inserir senhas, entre outras formas. Ao iniciar o lado do cliente aparecerá um menu contendo as funcionalidades principais da aplicação. Dependendo da escolha das funcionalidades, o cliente é levado para outro menu de login (podendo ser requerido o nome da conta(depósito) ou o nome e a senha(saque) da conta na mesma linha separados por uma vírgula).

Funcionalidades	Finalidade
Sacar	para saque de dinheiro de uma determinada conta
Depositar	para depósito de dinheiro de uma determinada conta
Abrir conta	para abertura de uma nova conta
Finalizar	finalizar a execução do cliente

Observações:

- ➔ O caixa funciona apenas com números inteiros;
- ➔ O menu de login não aparecerá caso a opção seja para criar uma conta.
- ➔ Ao criar a conta as informações devem ser separadas por vírgula e seguir o formato que se pede (nome,senha,saldo). Ex: “fulano,detal10,50”;
- ➔ A entrada deve seguir exatamente o que pede. Ex: no menu inicial ao digitar (1) para sacar um dinheiro, abrirá o login onde pedirá o nome e a senha da conta. O nome e senha devem ser digitadas separadas por uma vírgula, em outro caso não funcionará. *“fulano,detal10”*;

Mensagens

Criamos a classe Mensagem que organiza o formato das mensagens trocadas entre o lado do cliente e o lado do Servidor. Tem variáveis como o status da operação, qual operação está querendo ser feita e uma hash que armazena os possíveis parâmetros.

Status

Os status são:

Status	Significado
200	OK
373	SOLICITAÇÃO
400	ERROR
411	PARAMERROR

Operações

As operações para requisições são:

Operação	Função
LOGINSENHA	antes de **sacar** terá um login onde é requerido o **nome** e a senha da conta.
LOGINNOME	antes de **depositar** terá um login onde é requerido o **nome** da conta.
SAQUE	para **saque** de dinheiro de determinada conta
DEPOSITO	para **depósito** de dinheiro de determinada conta
CRIARCONTA	para **abrir conta**

Dificuldades encontradas

- ➔ **Uso das threads** - inicialmente tratamos de incluir threads apenas na funcionalidade de saque, para que a cada saque fosse criado uma thread.
- ➔ **Uso do envio e recebimento de dados Stream (*inputStream* e *outputStream*)** - Houve um tempo para entendermos bem do uso do Stream e seus tipos. De certa forma, criamos uma classe para tratamento dos dados, chamada *Mensagem*, que tem uma hash que age dinamicamente na adição de parâmetros.
- ➔ **Controle do saque** - O controle do saque tem a ver com o saque simultâneo e na proteção do dinheiro de cada um. Antes, como dito acima, criávamos uma thread para cada ato de sacar, mas depois vimos que tornar a criação da thread a cada requisição do cliente seria algo interessante, dessa forma criamos o *clientThread*, utilizando o exemplo disponibilizado pelo professor, e também "*syncronized*" para o controle de saques simultâneos da mesma conta.

Ideias

- ➔ Criação de uma API conectada a um banco de dados online(ex: API com nodeJS e mongoDB) para o substituir o arquivo .txt.
- ➔ Adição de algumas funcionalidades - edição de dados, exclusão de contas, help(para dúvidas), ver o saldo da conta
- ➔ Aumentar a segurança para proteção dos dados de cada conta.
- ➔ Melhorar o formato de dados do terminal como logins e abertura de contas, além de limitações relacionadas a entradas