ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

LICENCIATURA DE ENGENHARIA INFORMÁTICA / SEGURANÇA INFORMÁTICA EM REDES DE COMPUTADORES
SISTEMAS DISTRIBUÍDOS

Trabalho Prático

Miguel Barbosa - 8200102
Pedro Miguel – 8150206

2023/2024

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Índice

Introdução

Contextualização do problema

Este trabalho prático tem como foco aprimorar a comunicação nas Forças Armadas, permitindo a rápida transmissão de mensagens e instruções táticas. O desenvolvimento inclui a criação de um protocolo hierárquico, refletindo as patentes militares. A comunicação estende-se tanto entre entidades individuais quanto entre grupos, facilitando a coordenação. Além disso, a capacidade de criar canais específicos proporciona flexibilidade na disseminação de informações. O sistema visa garantir eficiência, segurança e controle hierárquico, contribuindo para operações mais eficazes e coordenadas nas Forças Armadas.

Objetivos

Os requisitos/objetivos estabelecidos são:

- Registo e autenticação de utilizadores
- Gestão de entidades
- Comunicação entre Clientes e Servidor
- Comunicação eficiente
- Canais de comunicação
- Gestão de autorização e pedidos
- Notificação e alertas
- Relatórios periódicos
- Capacidade do servidor
- Notificações multicast e Broadcast

Ferramentas utilizadas

Para o desenvolvimento do trabalho foram utilizadas as seguintes ferramentas

MySQL

O MySQL é responsável por gerir a base de dados relacional que guarda todas as informações necessárias de forma persistente.

Java

Este sistema foi desenvolvido com a linguagem de programação Java com recurso do IDE Netbeans.

Setup do sistema

- Instalação do sistema de gestão de base de dados MySQL.
- Instalação do Java 8.
- Executar classe DatabaseSetup.java.
- Executar classe Servidor.java (A base de dados tem que estar criada DatabaseSetup.java).
- Executar classe Main.java (O servidor tem que estar a correr Servidor.java).

Descrição do Sistema

Este sistema tem como base uma comunicação cliente-servidor através de comunicação com protocolos TCP. A opção deste protocolo deve-se ao facto de que na comunicação militar, a importância de que as mensagens cheguem ao destino é um dos fatores essenciais do sistema. Já para a comunicação dos grupos são utilizados grupos Multicast que utilizam o protocolo UDP pois a rápida comunicação é essencial para a comunicação entre grupos. Esta comunicação é feita através de pedidos enviados para ambos os lados com comandos de texto, que são interpretados e executados dependendo do comando e informação enviada.

Do lado do Cliente (Package Forças-Armadas) há 4 classes:

Classe Main - interface responsável por executar o sistema para a interação com o utilizador;

Classe Conectar - responsável pela conexão do cliente com o servidor;

Classe ConnectarGrupo - responsável pela conexão aos grupos Multicast e envio de mensagens para os mesmos.

Classe Server - responsável pela conexão dos grupos Multicast.

Do lado do Servidor (Package Servidor) há 10 classes:

Classe Central - Classe responsável pelo controlo da informação do servidor. É também responsável pelo armazenamento temporário de informação, como nº de utilizadores online no sistema bem como o nº de utilizadores online em cada grupo.

Classe Canal - define o modelo de dados Canal(porta, address(ip do servidor), nome) e contem métodos de controlo de dados da base de dados relativos ao canal.

Classe ConectarGrupo - responsável pela conexão aos grupos Multicast e envio de mensagens para os mesmos.

Classe DatabaseHelper - Contém métodos de controlo de dados da base de dados relativos a mensagens, utilizadores bem como a conexão com a base de dados.

Classe DatabaseSetup - Classe executável para criar a base de dados com todas as tabelas necessárias para o armazenamento de dados.(Entidades, Mensagens, Canais, Pedidos)

Classe Entidade - define o modelo de dados Entidade(id, username , password e cargo) e contém métodos de controlo de dados da base de dados relativos à entidade.

Classe EntidadeHandler - Entidade responsável para o envio de mensagens entre utilizadores, bem como algumas verificações, tais como se o utilizador é válido.

Classe Pedido - define o modelo de dados Pedido(id, user_criou_id, user_autorizou_id, user_aceitou_id, Mensagem do pedido e o estado) e contém métodos de controlo de dados da base de dados relativos aos pedidos.

Classe Server - responsável pela conexão dos grupos Multicast.

Classe Servidor - Classe executável que inicia o servidor. Responsável por receber os comandos enviados pelos utilizadores, envio de comandos para os mesmos, bem como o envio das mensagens/notificações periódicas para os utilizadores e grupos. O Servidor é iniciado usando ServerSockets. Esta classe utiliza o protocolo de comunicações TCP, pois a garantia de que as mensagens chegam ao destino é um dos pontos essenciais do sistema. O servidor utiliza Unicast para comunicar com os utilizadores quando realizam pedidos através de comandos, Broadcast quando é emitido um alerta e é enviado uma mensagem para todos os utilizadores conectados (a mensagem é também enviada e guardada para todos os utilizadores que estão offline) e Multicast para as mensagens de canais (Grupos Multicast) e para as notificações periódicas.

Documentação do utilizador

/registo ‘user’ ‘pass’ ‘rank’ - Permite registar um novo utilizador (rank: Capitao, General ou Sargento);

/login ’user’ ‘pass’ - Permite autenticar no sistema;

/mensagens - Permite ler todas as mensagens do utilizador;

/enviar ‘user’ ‘mensagem’ - Permite enviar uma mensagem a um utilizador;

/grave ‘mensagem’ - Envia um alerta para todos os utilizadores

/canais - Apresenta todos os canais disponíveis, e se está conectado ou não;

/canal ‘porta’ - Permite alterar se está conectado ou não a um canal. (Necessita de fazer logout e login para conectar)

/canalenviar 'numero' - Permite enviar uma mensagem para o grupo com a porta introduzida;

/canaladd ‘porta’ - Permite adicionar um novo canal (apenas para o cargo Capitão);

/canaldel ‘porta’ - Permite remover um canal (apenas para o cargo Capitão);

/pedidos - Mostra todos os pedidos e o estado dos mesmos;

/realizar 'pedido' - Permite criar um novo pedido;

/autorizar ‘id’ - Permite autorizar um pedido (Dependendo do cargo);

/aceitar ‘id’- Permite aceitar um pedido;

/exit - volta ao menu anterior;

Comandos

No lado do cliente há o comando /login .

Quando o cliente inicia a aplicação cria uma nova conexão com o servidor e utiliza 2 Threads. 1 thread para enviar mensagens ao Servidor e outra para receber mensagens do Servidor. Quando o utilizador faz o login, o Servidor envia ao cliente através do seu socket, um comando /login 'portas' que deve começar por /login e seguido pelas portas dos grupos de multicast a que o utilizador pertence. Quando o cliente completa o processo de login, é adicionado à lista de sockets do servidor para receber mensagens (Unicast e Broadcast). Por sua vez o cliente com as portas que recebeu do servidor dos seus grupos, entra automaticamente nos mesmo.

No lado do servidor temos os seguintes comandos:

/registo	/canal
/login	/sairgrupo
/logout	/pedidos
/mensagens	/realizar
/enviar	/autorizar
/canais	/aceitar
/canaladd	/grav
/canaldel	

/registo ‘username’ ‘password’ ‘Cargo’

Quando o cliente envia o comando /registo username password Cargo, é enviada uma String para o servidor com este mesmo formato, o servidor interpreta o comando /registo e divide a mensagem em 4 partes. Depois faz a verificação das partes tais como se os dados introduzidos são válidos

A autenticação é feita pela função ‘loginEntidade’ solicitando o nome e a *password* verificando se correspondem às informações armazenadas no sistema, se o *login* for bem-sucedido a variável ‘isLogin’ passa a ser *true* indicando que o usuário está autenticado.

/login

Quando o cliente envia o comando /login username password, é enviada uma string para o servidor com esse formato, a função divide a mensagem recebida e armazena as partes em um array e verifica se o número de partes é 3. Se a mensagem tem 3 partes, a função extrai o nome de utilizador e a password da segunda e terceira parte, respetivamente.

Verifica se tanto o nome e a password não são nulos, obtém uma entidade com base no nome de utilizador , e se bem-sucedido, é enviado ao utilizador através do ‘writer’ o nome e a lista de portas associadas aos canais da entidade.

/logout

Quando o cliente faz logout, termina a conexão com os grupos (que conectou ao fazer login), e envia um comando /logout ao servidor, que por sua vez remove o utilizador da lista de sockets. Apos o utilizador fechar a aplicação (Main.java) é terminada a conexão (Threads para envia e receber mensagens).

/mensagens

Quando o cliente envia o comando /mensagens, envia ao Servidor, através do socket (TCP - Unicast), uma String com "/mensagens 'utilizador'.O Servidor separa a String para obter o nome de utilizador, verifica se o utilizador existe, se tem mensagens e fazer o pedido à base de dados com as mensagens deste utilizador. As mensagens são enviadas ao utilizador através do socket (TCP - Unicast).

/enviar

Quando o utilizador envia o comando /enviar 'user' 'mensagem', envia ao Servidor, através do socket (TCP - Unicast), uma String com "/enviar 'user' 'mensagem' 'user'.O Servidor separa a String para obter o nome do utilizador que envia a mensagem, o utilizador destinatário da mensagem e a mensagem. Verifica se os utilizadores existem e guarda a mensagem na base de dados para o utilizador destinatário da mensagem. Uma mensagem de envio com sucesso é enviada ao utilizador após o envio da mensagem através do socket (TCP - Unicast).

/canais

Quando o utilizador envia o comando /canais, envia ao Servidor, através do socket (TCP - Unicast), uma String com "/canais 'user'. O Servidor separa a String para obter o nome do utilizador que envia a mensagem. Verifica se o utilizador existe. Se o utilizador existe, é feito um pedido à base de dados com todos os canais criados e enviados ao utilizador através do socket (TCP - Unicast). É apresentado ao utilizador se este pertence aos respectivos grupos ou não.

/canaladd

Quando o utilizador envia o comando /canaladd ‘porta’ ‘nome’, envia ao Servidor, através do socket (TCP - Unicast), uma String com “/canaladd ‘porta‘ ‘nome’. Este comando cria um novo canal com base na mensagem de entrada. Verifica se o criador do canal possui o cargo de Capitão e, em caso possuía o cargo, cria um novo canal com informações extraídas da mensagem. O novo canal é adicionado à instância da classe “central”.

/canaldel

Quando o utilizador envia o comando /canaldel ‘porta, envia ao Servidor, através do socket (TCP - Unicast), uma String com “/canaldel ‘porta‘. Usando o comando /canaldel é possível remover um canal com base na mensagem de entrada. Verifica se o utilizador possui o cargo de Capitão e, em caso afirmativo, remove o canal.

/canal

Quando o utilizador envia o comando /canal ‘porta’, envia ao Servidor, através do socket (TCP - Unicast), uma String com “/canal ‘porta’ ‘user’. O Servidor separa a String para obter o nome do utilizador que envia a mensagem e a porta. Verifica se o utilizador existe. Se o utilizador existir e o grupo existir, atribui o canal ou remove o canal atribuído ao utilizador. É enviado ao utilizador através do socket (TCP - Unicast) uma mensagem de sucesso na operação.

/pedidos

Quando o utilizador envia o comando /pedidos, envia ao Servidor, através do socket (TCP - Unicast), uma String com "/pedidos 'user'. O Servidor separa a String para obter o nome do utilizador que envia a mensagem. Verifica se o existe algum pedido. Se existir pedidos, é feito um pedido à base de dados com todos os pedidos e são enviados ao utilizador com os dados (id, utilizador que criou, utilizador que autorizou, utilizador que aceitou, mensagem do pedido e o estado, se aplicável ao pedido) através do socket (TCP - Unicast).

/realizar

Quando o utilizador envia o comando /realizar 'pedido', envia ao Servidor, através do socket (TCP - Unicast), uma String com "/realizar 'pedido'. O Servidor separa a String para obter o nome do utilizador que envia a mensagem. Verifica se o utilizador existe. Se o utilizador existir, é criado um novo pedido no estado criado (aguarda autorização) e adicionado à base de dados com o utilizador que o criou atribuído no estado de criado. É apresentado ao utilizador uma mensagem de sucesso através do socket (TCP - Unicast). É enviada uma mensagem (Broadcast) para todos os utilizadores que um novo pedido foi criado.

/autorizar

Quando o utilizador envia o comando /autorizar 'id do pedido', envia ao Servidor, através do socket (TCP - Unicast), uma String com "/autorizar 'id do pedido' 'user'. O Servidor separa a String para obter o nome do utilizador que envia a mensagem. Verifica se o utilizador e o pedido existe. Verifica se o pedido está no estado de aguardar autorização. Verifica se o utilizador tem permissão para autorizar o pedido. Se tudo isto for verificado, o pedido passa para o estado autorizado e alterado na base de dados com o utilizador que o autorizou atribuído. É apresentado ao utilizador uma mensagem de sucesso através do socket (TCP -

Unicast). É enviada uma mensagem (Broadcast) para todos os utilizadores que um pedido foi autorizado.

/aceitar

Quando o utilizador envia o comando /aceitar ’id do pedido’, envia ao Servidor, através do socket (TCP - Unicast), uma String com “/aceitar ‘id do pedido’ ‘user’. O Servidor separa a String para obter o nome do utilizador que envia a mensagem. Verifica se o utilizador e o pedido existe. Verifica se o pedido está autorizado. Se tudo isto for verificado, o pedido passa para o estado autorizado e alterado na base de dados com o utilizador que o aceitou é atribuído. É apresentado ao utilizador uma mensagem de sucesso através do socket (TCP - Unicast). É enviada uma mensagem (Broadcast) para todos os utilizadores que um pedido foi autorizado.

/grave

Quando o utilizador envia o comando /grave ‘mensagem’, envia ao Servidor, através do socket (TCP - Unicast), uma String com “/grave ‘mensagem’ ‘user’. O Servidor separa a String para obter o nome do utilizador que envia a mensagem e a mensagem. Verifica se o utilizador e o pedido existem. É enviada uma mensagem (Broadcast) para todos os utilizadores com a mensagem de alerta e quem enviou o alerta.

Conclusão

Neste trabalho utilizou-se todos os conhecimentos obtidos durante as aulas tais como: Protocolos UDP e TCP, sockets etc. Este trabalho enriqueceu-nos sobre os diferentes protocolos e como/quando devem ser usados.