RELATÓRIO IRC

Implementação de um servidor de email

• U  C •

Realizado por:

Teresa Salazar 2015234237 | Gonçalo Amaral 2015249122

INTRODUÇÃO

O objetivo deste trabalho é implementar um servidor de email para múltiplos clientes em simultâneo. A comunicação entre o cliente e o operador suporta diversas primitivas através do uso de sockets. O sistema suporta um operador que tem permissão para alterar a base de dados. O projeto foi realizado em python.

SETUP DE SERVIDOR E CLIENTES

Na pasta **server**, após correr o comando *python server.py –p PORT*, o servidor cria uma socket utilizando a função *socket.socket()*. Posteriormente, através da função *bind()* é associado um socket a um determinado endereço. Com *listen()* é iniciada a conexão utilizando o protocolo TCP. Depois, o servidor vai estar num ciclo constante a aceitar novos clientes através da função *accept()* e da utilização de threads.

Na pasta **client,** após correr o comando *python client.py –p PORT*, o client cria uma socket e conecta-se ao servidor. Desta forma, é possível enviar e receber dados do servidor.

Posteriormente, é solicitado ao cliente que escolha uma opção do menu.

REGISTO

Se o cliente escolher a opção '0', é primeiro enviado ao servido a respetiva opção através da função *send()* que permite transmitir uma mensagem em TCP. O servidor, por sua vez, irá receber a respetiva opção através da função recv(), que permite receber uma mensagem TCP.

Após introdução de username (com a verificação do comprimento ter de ser menor do que 9 carateres) e da password por parte do cliente, é enviado ao cliente esta informação. O servidor, depois de encriptar a password utilizando o módulo *hashlib* do python, escreve no ficheiro "clients.txt" o novo user. É desta forma terminado o registo do novo utilizador e o cliente é redirecionado para o menu novamente.

LOGIN

Se o cliente escolher a opção '1', é lhe pedido que introduza o username e a password. Depois é enviada para o servidor esta informação. Através do ficheiro "clients.txt" o servidor consegue determinar se o cliente é válido e autorizado. Posteriormente, o servidor informa o cliente se o user é válido enviando novamente através da função *recv()* um '0' ou um '1'.

LISTAR CLIENTES AUTORIZADOS

Se o cliente escolher a opção '3', após enviar informação sobre a opção pretendida através da função *send()*, o servidor irá percorrer o ficheiro "clients.txt" e juntar numa string todos os usernames. Esta string depois será enviada ao cliente e imprimida na sua consola.

ENVIAR MENSAGEM

Se o cliente selecionar a opção '4', é lhe pedido que escreva a o nome do cliente e o texto da mensagem. O servidor, por sua vez, depois de receber o nome do cliente e a mensagem, verifica se o cliente é válido e se está no ficheiro "clients.txt". Caso isto se confirme é escrita no ficheiro "messages.txt" informação sobre: quem enviou a mensagem, o texto da mensagem, o recetor da mensagem e um "0". Este "0" significa que o cliente ainda não leu a mensagem.

Finalmente, é enviada uma notificação ao cliente recetor da mensagem caso este esteja ligado ao servidor. Se ele não estiver ligado ao servidor, a notificação só é enviada quando ele fizer o login.

LISTAR MENSAGENS

O cliente tem duas opções: listar mensagens lidas (opção '5') ou mensagens não lidas (opção '2'). Em ambas opções, caso o cliente não tiver nenhuma mensagem é lhe transmitida essa informação. Se o cliente tiver mensagens lidas ou não lidas, o servidor abre o ficheiro "messages.txt" e verifica quais são as suas mensagens. As mensagens não lidas estão identificadas com um "0", enquanto que as mensagens lidas estão marcadas com um "1". Desta forma, conforme o pedido do cliente, o servidor pode-lhe enviar o tipo de mensagens desejado. Quando o cliente pede para ler novas mensagens, estas serão depois marcadas com o "1".

APAGAR MENSAGEM

Se o cliente selecionar a opção '6' e este tiver feito o login ou o registo, é listado ao cliente um conjunto de todas as suas mensagens e do respetivo iD. Após o cliente introduzir o iD da mensagem que quer apagar, o servidor apaga do ficheiro "messages.txt" a mensagem. Não é permitido ao cliente alterar mensagens que não sejam suas.

ALTERAR PASSWORD

Se o cliente selecionar a opção '7', é lhe pedido que escreva a nova password. Esta é enviada para o servidor e lá é codificada. Assim, no ficheiro "clients.txt" é procedida a substituição da password original pela nova password.

MODO OPERADOR

Se o cliente selecionar a opção '8', o servidor, após receber a opção pretendida, verifica se o utilizador é o operador. Este serviço permite apenas um operador, cuja identificação está no ficheiro "superuser.txt". Caso o cliente não seja operador é lhe perguntado se deseja obter esses privilégios. Se sim, é lhe pedido que digite a password do operador. Caso esta esteja correta, o cliente passa a ser o operador.

O operador tem o seu próprio menu. A opção '1' permite-lhe retirar um cliente da base de dados (ficheiro "clients.txt"). A opção '2' permite-lhe retirar uma mensagem da base de dados (ficheiro "messages.txt"). A opção '3' retorna o cliente para o menu principal.

ABANDONAR O SISTEMA

Se o cliente selecionar a opção '9' é fechada a socket através da função *close()*. É também imprimido na consola do cliente a informação do cliente que saiu do sistema.

CONCLUSÃO

Concluindo, acreditamos que cumprimos tudo o que era pedido no trabalho de forma eficaz e eficiente. Para além disso, a comunicação entre o cliente e o servidor é bastante "user friendly". Devido ao código estar bem protegido, é praticamente impossível a aplicação deixar de funcionar. É de salientar que fomos além do pedido no trabalho, implementando funcionalidades entra como o "registo" e algumas funcionalidades do operador.

Para terminar, com este trabalho aprendemos a trabalhar melhor com sockets com outra linguagem que não C.