UNIVERSIDADE DE COIMBRA

FACULDADE DE CIÊNCIAS E TECNOLOGIA

MESTRADO INTEGRADO EM
ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

Redes de Computadores

Projeto: Servidor de mensagens escritas

Fabian Pascual Dias

2016107423

João Vitor Sgotti Veiga

2017170653

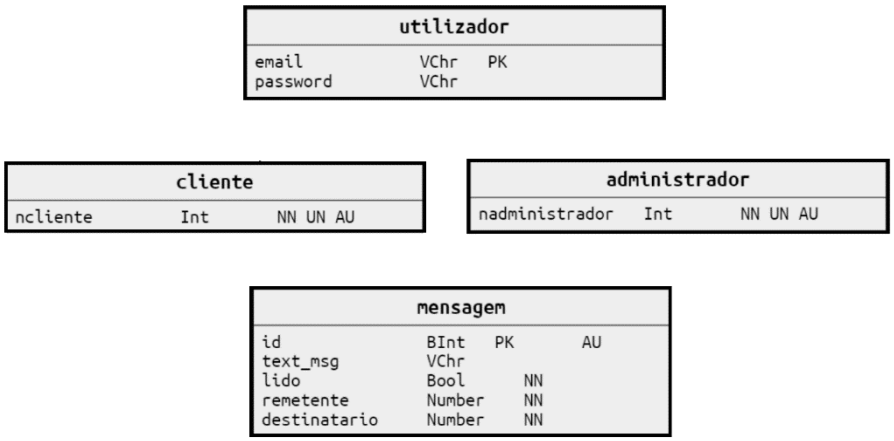
31 de Maio de 2020

1. Introdução

O objetivo do trabalho foi que implementassemos um sistema similar a um servidor de email, feito por ligações TCP. Onde fosse possível gerir a lista de cliente, isto é inserir e apagar clientes, usando o acesso exclusivo de administrador.
Para estes clientes, seria possivel que enviassem mensagens para um ou vários outros clientes, que tivessem uma caixa de mensagens, classificada em mensagens lidas e mensagens não lidas. Posteriormente a leitura de uma mensagem seria possivel apaga-la da caixa de entrada.

No projeto apresentado, foi possível implementar todas as funcionalidades descritas acima. Alem de funcionalidades extras: uma base de dados integrada, para que houvesse mais confiabilidade no armazenamento das mensagens. Estas serão salvas no servidor **PostgreSQL**, desenvolvido pela University of California, em Berkeley e atualmente o mais utilizado em todo o mundo. Também foi implementada a função "validação", que traz uma robustez contra erros de digitação do utilizador, evitando que o programa aceite um valor desconhecido e seja corrompido. Para isto, foi escolhido usar a liguagem de programação **Python 3.7**, compilada através do **PyCharm**.

1. Diagrama Físico da Base de dados



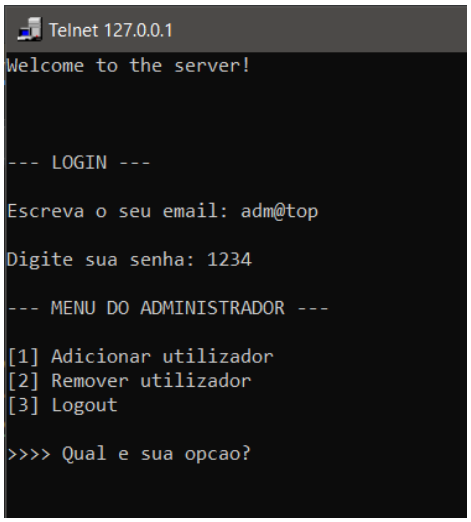
Para a fundamentação da base de dados, usou-se a ferramenta **onda**, criada pelo departamento de informática da Universidade de Cimbra. A partir deste diagrama físico, pode-se gerar um script **(em anexo)**

2. Organização da Aplicação

A Aplicação é dividida entre três sub-interfaces (Login, Menu Administrador e Menu Cliente), todas dispostas no mesmo file main. Entretanto, todas dispõe de funcionalidades atômicas, isto é distintas e não compartilhadas entre si.

2.1 Login

Nela distingui-se os usuários através de suas credenciais (login e password) em dois grupos, administradores e clientes, na qual serão direcionados para os seus menus específicos (ou caso as credenciais não sejam válidas, será impressa uma mensagem de erro e estas serão novamente requeridas), caso seja a identificação do cliente "ncliente" será atribuida a todas as funçoes subsequentes.

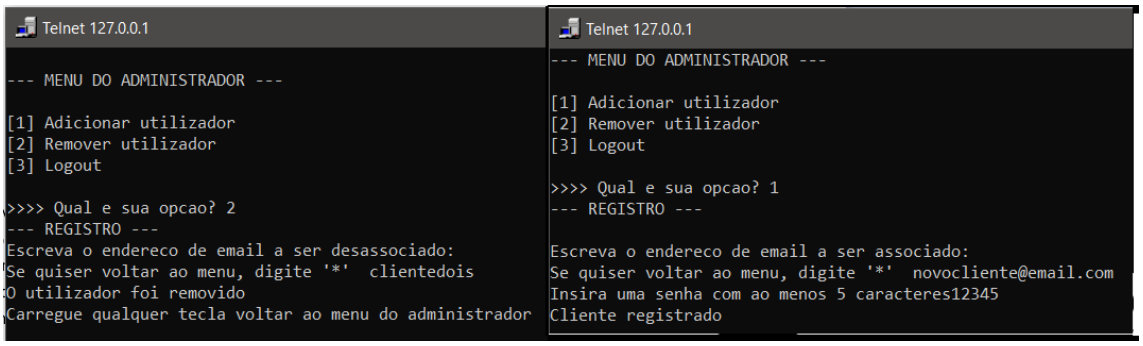


2.2 Menu Administrador

I – Adicionar Utilizador: Onde o administrador pode adicionar um novo cliente através de email e senha único.

II – Remover Utilizador: Permite ao administrador remover um cliente, através de seu email

III – Logout: Onde o administrador poderá sair da sua conta, voltando ao menu Login



2.3 Menu Cliente

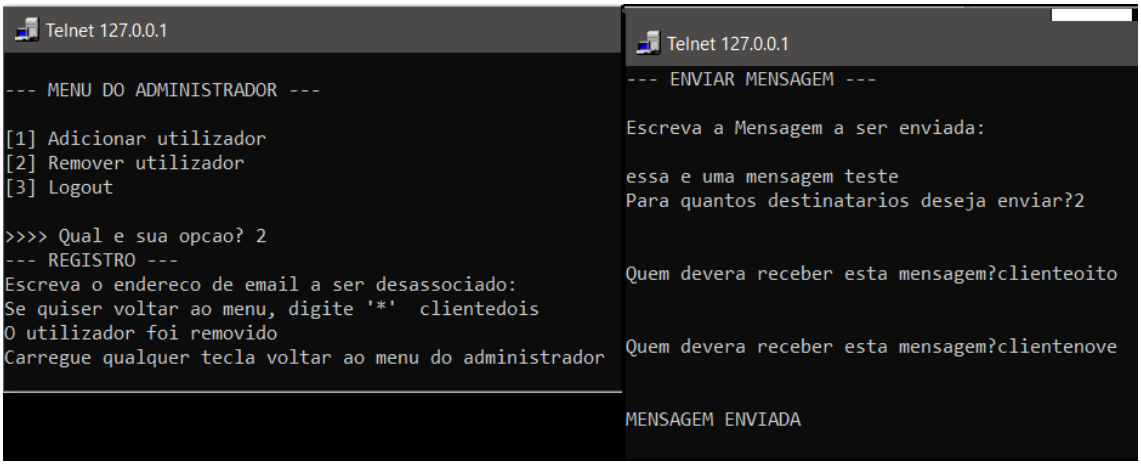
I- Consultar Mensagens:

Permite ao cliente efetuar a leitura de mensagens enviadas pelos outros clientes, isto é, que constem na sua caixa de entrada individual. Para mais, estas mensagens podem ser divididas entre as lidas e as ainda não lidas, de acordo com um parâmetro *booleano* na primeira entidade mencionada por cada respectivo cliente. Também, nesta opcão é possivel apagar mensagem lidas.

II- Enviar Mensagens:

Permite ao cliente enviar mensagem a outros utilizadores, tendo a opção de envio do mesmo corpo de texto para quantos outros clientes desejar, inclusive para sí próprio, assim como os servidores de email.

III- Logout: Onde o administrador poderá sair da sua conta, voltando ao menu Login



3. Comunicação

3.1 Servidor

Ao utilizarmos a linguagem python devido a sua facilidade de manipular dados e ponteiros, acabou assim nos permitindo um código mais fluido e legível. Tendo o esforço inicial para aprender a linguagem foi compensatório uma vez que se mostrou muito eficiente para a implementação da base de dados a manipulação das sockets e também a conexão do servidor que passa a ser mais intuitiva.

Para criarmos o servidor associamos a uma socket, a utilização dos protocolos IPV4 com uma conexão TCP. Em seguida ligamos a socket ao IP e o PORT desejados e através da função "listen()", o servidor fica a espera de novas conexões. Para verificarmos que tudo está a funcionar o servidor imprime "on-line" no ecrã .

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(("127.0.0.1", 1234))
s.listen()
print("on-line \r\n")

#executa o programa
```

3.2 Cliente

Inicialmente planejamos criar duas funções, client e server, mas após uma reestruturação da abordagem decidimos seguir a recomendação do enunciado do projeto e utilizar a ferramenta telnet. Devido uma dificuldade em encontrar uma documentação que explicasse o funcionamento do telnet, tivemos que efetuar vários testes para percebermos o seu real funcionamento. A maior dificuldade foi obter uma string através do programa, pois o servidor recebia um stream constante de caracteres seguidos por "\r\n" que eram inseridos no terminal do telnet. Para contornarmos esse problema criamos um ciclo que captura estes caracteres modificando-os e adicionando em uma tabela de formato string até que o cliclo indentifique um "enter"

3.3 Inputs e Outputs

Para uma integração elegante entre o código da base de dados e da comunicação entre o cliente foram criadas duas funções, "ask" e "mostra".

A função ask inspirada na função input tem como objetico enviar uma mensagem para o cliente e obter uma resposta utilizando o ciclo previamente mencionado para captura de dados. Tendo como parâmetros de entrada uma string e o socket do cliente, para que a mensagem seja enviada para o devido utilizador. E ao ser executada retorna a string inserida pelo cliente.

```
def ask(frase,client):
    message = ""
    client.send(bytes(frase, "utf-8"))
    while True:
        resp = client.recv(20)
        x = resp.decode("utf-8")
        if(x == "\r\n"):
            return message
        message = message + x
```

A função “mostra” tem a mesma funcionalidade da função print, assim apenas enviando uma mensagem para o cliente. Com os mesmos parâmetros da função ask (frase e cliente)

```
def manda(frase,client):
    client.send(bytes(frase+"\r\n","utf-8"))
```

3.4 Codificação

Atualmente a “utf-8” representa 95% das codificações utilizadas na internet. Tal sucesso vem da sua compatibilidade com a tabela ASCII assim tendo acesso a quase todas as linguagens de programação. Por ser uma codificação de caracteres de tamanho variável, tem capacidade de 4bytes (32bits). Devido não possuir caracteres especias (ç, ~, ^, ´, `, entre outros) o programa acaba tendo que sofrer alterações na escrita, tendo como sugestão para evitar erros e contornar o problema o cliente pode escrever as suas mensagens em inglês.

4. Disposições Gerais

O projeto consumiu aproximadamente 50 horas de esforço por aluno.
Mesmo sabendo que a implementação através de base de dados seria mais demorada e complexa, optamos por este caminho, por acreditar ser mais edificante para nossas habilidades como programadores, lidar com situações novas que demandem pesquisa e novos aprendizados, além de que, a implementação através da base de dados dá mais segurança e estabilidade ao programa, pois não há riscos de perder os registros das mensagens.

5. Bibliografia

https://en.wikipedia.org/wiki/UTF-8
https://stackoverflow.com/
https://pt.wikipedia.org/wiki/PostgreSQL