FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Departamento de Engenharia Informática

Ano Lectivo de 2018/2019

Introdução às Redes e Comunicação

IoT Student Advisor and Best Lifestyle Analyzer

(ISABELA)

António Marques Maria - 2017265346

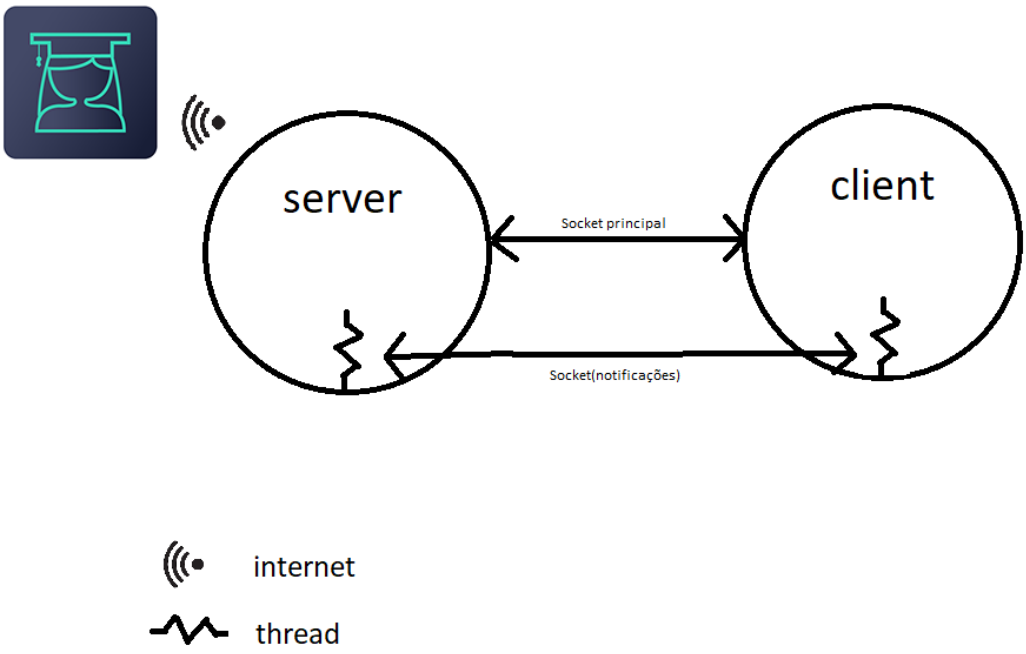
David Jesus Vaz Cortesão - 2008109004

Introdução

Linguagem escolhida - C

A comunicação efetuada entre o cliente e o servidor irá ser efetuada através de dois sockets, um socket para efetuar as operações desejadas entre o cliente e o servidor, tais como, visualizar dados dele próprio, dados do grupo, e subscrever a notificações caso algum dos dados do grupo altere. O segundo socket irá servir para enviar e receber essas mesmas notificações que irão ser escritas e lidas por threads dos respetivos processos.

O server também irá buscar as informações relativamente aos clientes à API da ISABELA.



A ligação à ISABELA é efetuada através do protocolo HTTP enquanto que as ligações entre server e cliente são feitas através do protocolo TCP.

PRIVACY SERVER

O nosso servidor fica à espera que algum cliente se conecte a ele através do socket 'fd'.

```
int fd, client;
struct sockaddr_in addr, client_addr;
int client_addr_size;

//     cria estrutura
bzero((void *) &addr, sizeof(addr));

// Configure settings of the server address struct
// Address family = Internet
addr.sin_family = AF_INET;
//Set IP address to localhost
addr.sin_addr.s_addr = htonl(INADDR_ANY);
//Set port number, using htons function to use proper byte order
addr.sin_port = htons(SERVER_PORT);

//Create the socket.
if ( (fd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    erro("na funcao socket");
//Bind the address struct to the socket
if ( bind(fd,(struct sockaddr*)&addr,sizeof(addr)) < 0)
    erro("na funcao bind");
//Listen on the socket, with 5 max connection requests queued
if( listen(fd, 5) < 0)
    erro("na funcao listen");

client_addr_size = sizeof(client_addr);
client = accept(fd,(struct sockaddr *)&client_addr,(socklen_t *)&client_addr_size);
```

Neste excerto de código é criado o socket e estabelece-se a ligação com os clientes.

Depois de aceitar a ligação o host vai criar um processo para processar os pedidos do cliente.

Cada cliente é capaz de visualizar a sua própria informação, e relativamente aos grupos, é capaz de ver a informação que não for privada, por exemplo, não irá conseguir visualizar o ID das pessoas do grupo, a localização de cada uma, as atividades e os departamentos.

SUBSCRIÇÕES E NOTIFICAÇÕES

As subscrições são feitas através duma função que quando o cliente quer subscrever a uma informação específica ou a todas ele altera o valor de um boolean que está criado na base de dados do server. As subscrições podem ser feitas de um dado especifico ou de todos e podem ser alteradas.

As notificações funcionam através de uma thread que compara os dados antigos do grupo com os atualizados, segundo a segundo. Se os valores forem diferentes ele atualiza a estrutura antiga do grupo e continua a rotina. Enquanto faz isto, se o valor da subscrição for TRUE ele envia uma mensagem para o thread criado no cliente, que está constantemente à espera de uma mensagem do socket(das notificações), a dizer que esse dado especifico foi alterado.

As threads também são responsáveis por criar e estabelecer a ligação entre si.

```
while(done==0){
    calcula_media();
    if(ogrupo.calls_recebidas!=grupo.calls_recebidas){
        ogrupo.calls_recebidas=grupo.calls_recebidas;
        write(noti_fd,"Entrou aqui 2\n",BUF_SIZE);
        if(total_pessoas[i].sub_call_recebidas==true){
            write(noti_fd,"O valor das chamadas recebidas foi alterado!",BUF_SIZE);
        }
    } else if(ogrupo.calls_feitas!=grupo.calls_feitas){
        ogrupo.calls_feitas=grupo.calls_feitas;
        if(total_pessoas[i].sub_call_feitas==true){
            write(noti_fd,"O valor das chamadas feitas foi alterado!",BUF_SIZE);
        }
    } else if(ogrupo.calls_duracao!=grupo.calls_duracao){
        ogrupo.calls_duracao=grupo.calls_duracao;
        if(total_pessoas[i].sub_call_duracao==true){
            write(noti_fd,"O valor da duração das chamadas foi alterado!",BUF_SIZE);
        }
    } else if(ogrupo.calls_perdidas!=grupo.calls_perdidas){
        ogrupo.calls_perdidas=grupo.calls_perdidas;
        if(total_pessoas[i].sub_call_perdidas==true){
            write(noti_fd,"O valor das chamadas perdidas foi alterado!",BUF_SIZE);
        }
    } else if(ogrupo.sms_recebidas!=grupo.sms_recebidas){
        ogrupo.sms_recebidas=grupo.sms_recebidas;
        if(total_pessoas[i].sub_sms_recebidas==true){
            write(noti_fd,"O valor das SMS recebidas foi alterado!",BUF_SIZE);
        }
    } else if(ogrupo.sms_enviadas!=grupo.sms_enviadas){
        ogrupo.sms_enviadas=grupo.sms_enviadas;
        if(total_pessoas[i].sub_sms_enviadas==true){
            write(noti_fd,"O valor das SMS enviadas foi alterado!",BUF_SIZE);
        }
    }
    sleep(1);
}
```

Rotina do thread server.

```
while(done==0){
    nread = read(fd2, buffer, BUF_SIZE);
    buffer[nread] = '\0';
    printf("%s\n", buffer);

}
```

Rotina do thread cliente.