IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

Trabalho Prático

Aplicações de Redes de Computadores

Departamento de Sistemas e Informática

Manual Técnico

Empresa ParquesSado

Docente:
Rossana Santos

Realizado por:
Luís Trindade nº080221028 Turma 2
Samuel Soares nº080221076 Turma 2

2010/2011

Índice

Introdução

No âmbito da disciplina de Aplicações em Redes de Computadores, foi-nos proposto a realização de um trabalho prático. Este trabalho prático tinha como objectivo a realização de um sistema para a gestão da empresa ParquesSado que é responsável pela exploração e manutenção de locais de estacionamento público.

O manual técnico consiste em demonstrar todos os passos que foram feitos para a realização do projecto pedido.

Através de diagramas temporais vamos mostrar os processos desenvolvidos para uma melhor compreensão da comunicação entre eles.

Em primeiro lugar irá ser feita uma breve apresentação da aplicação desenvolvida, bem como as justificações de todas as opções de implementação. Irá ser apresentada a arquitectura do projecto e uma descrição pormenorizada e especifica dos protocolos de comunicação desenvolvidos ao nível da aplicação. Por fim serão apresentados as limitações do programa.

Apresentação da aplicação

A empresa ParquesSado é responsável pela gestão, exploração e manutenção de locais de estacionamento público. É também parceira do Ministério das Finanças.

Possui, neste momento, a concessão do estacionamento público de diversas cidades de Portugal (existe uma agência da empresa em cada uma e a sede é em Setúbal) sendo a exploração destes espaços a sua principal fonte de rendimento.

Para que a exploração seja mais eficiente a empresa necessita de um sistema que permita aos seus funcionários no terreno comunicar com a empresa e obter toda a informação necessária para cumprirem as suas funções.

Pretende-se implementar um sistema de comunicação distribuído que será utilizado no controlo de estacionamento abusivo e no controlo de imposto de circulação em falta.

Como se têm verificado muitos casos de falsificação de cartões de residente, a empresa decidiu abolir os cartões de residente. Para tal, os seus fiscais têm de validar se um dado carro está correctamente estacionado fornecendo a matrícula ao sistema que de seguida informa se existe alguma infracção.

Um dos problemas em não existirem cartões de residente é o do proprietário do carro poder não se aperceber da data de renovação da autorização de estacionamento e ser multado pelo prazo ter terminado. O sistema deverá minimizar este problema.

Justificação de todas as opções implementadas

A aplicação baseia-se em três componentes:

- Servidor da Sede

- Servidor da Cidade

- Funcionário (Cliente)

Servidor Sede

Escolhemos uma arquitectura semelhante à que fizermos nos laboratórios, a Classe Main, a classe TrataCliente e uma classe Funcionário.

Na classe Main temos um ciclo infinito que cria uma thread com o método trataLigacao da classe TrataCliente.

Na classe TrataCliente recebe a ligação de um Servidor Cidade e temos um array de funcionários que é utilizado para comparação no login.

Na classe Funcionario temos um nome, uma cidade e uma zona para identificação do funcionário.

Servidor Cidade

Escolhemos uma arquitectura semelhante à que fizermos nos laboratórios, a classe TrataCliente e a classe Matricula.

Na classe Main ocorre a ligação do Servidor Cidade ao Servidor Sede e recebe ligações do cliente.

Na classe TrataClientes temos um array de matrículas, a verificação do login, as mensagens recebidas do funcionário para a diferenciação entre mensagens, matriculas e ordens, e envia mensagens para os funcionários que estão ligados à cidade.

Na classe Matricula temos uma zona e uma matrícula para identificação de uma matrícula.

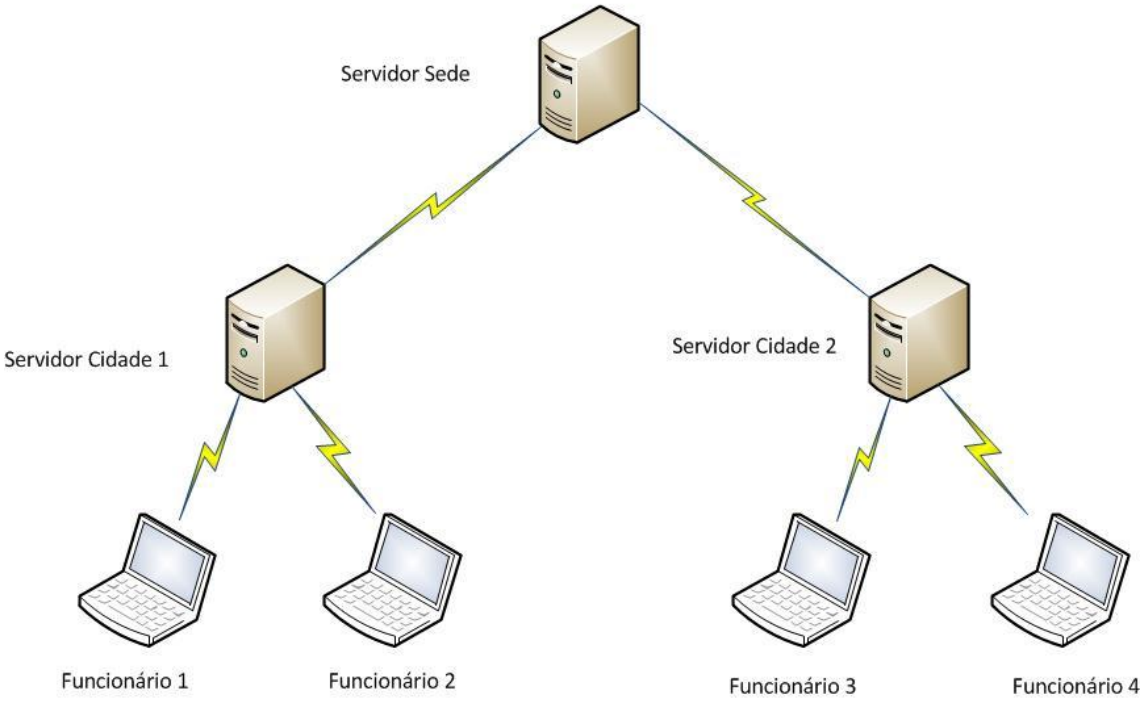
Funcionário

No Funcionario temos uma classe Main e um ThreadCom.

Na classe Main chama-se a classe ThreadCom.

Na classe ThreadCom temos duas threads, a reader e a writer que tratam de enviar e receber as mensagens trocadas com o Servidor Cidade.

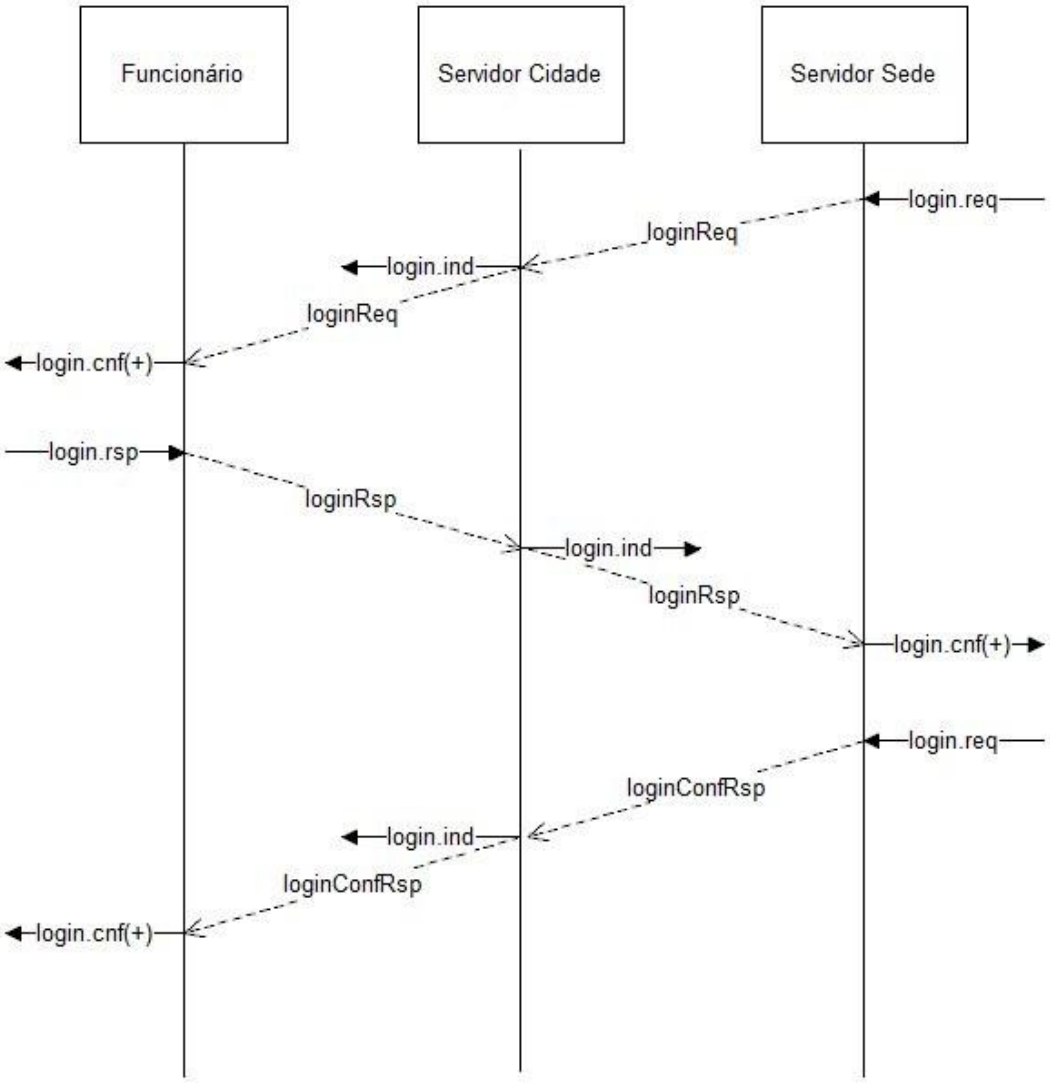
Arquitectura da solução



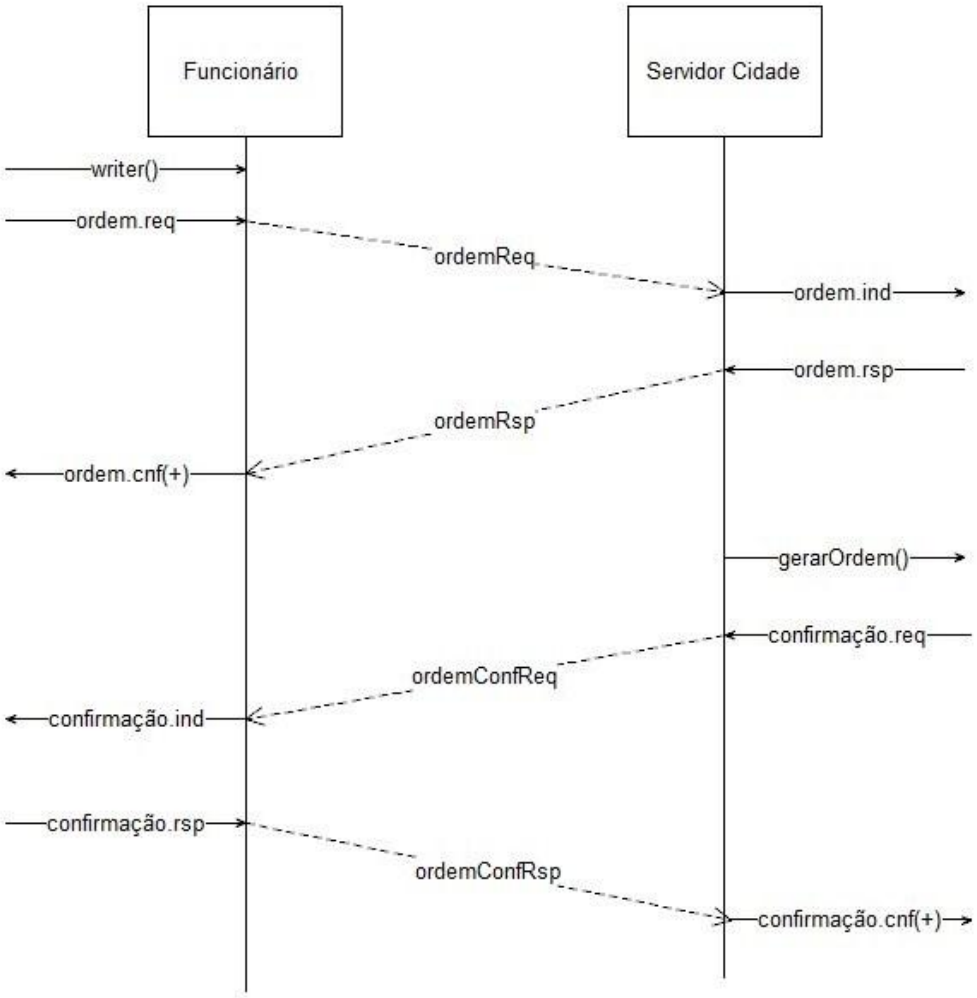
Legenda		
Subtítulo da Legenda		
Símbolo	Contagem	Descrição
	3	Servidor
	6	Ligação de comunicações
	4	Computador portátil

Diagramas temporais

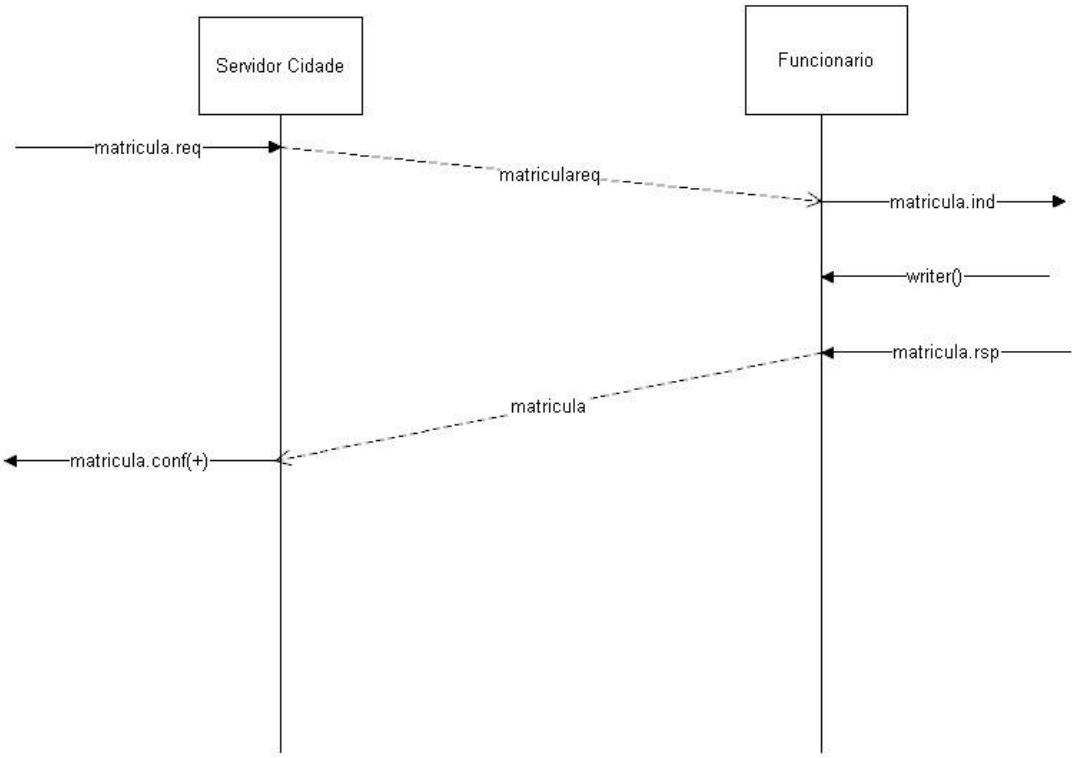
Login



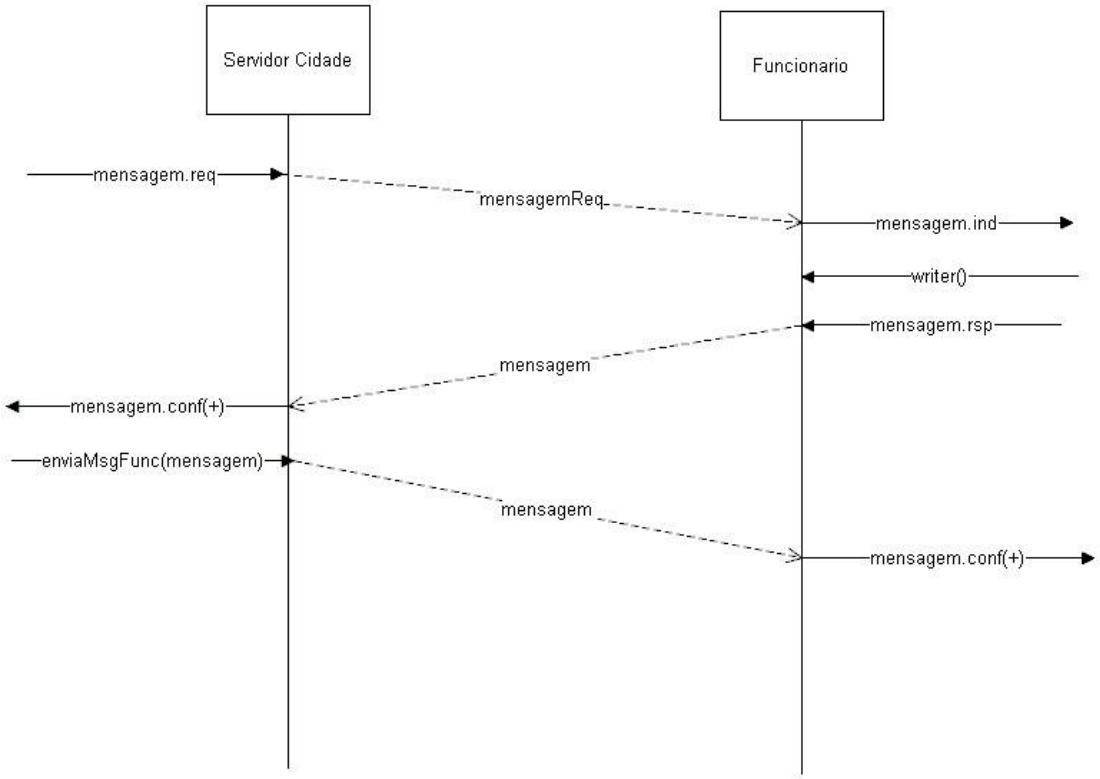
Ordem


Funcionário
Servidor Cidade
writer()
ordem.req
ordemReq
ordem.ind
ordem.rsp
ordemRsp
ordem.cnf(+)
gerarOrdem()
confirmação.req
ordemConfReq
confirmação.ind
confirmação.rsp
ordemConfRsp
confirmação.cnf(+)

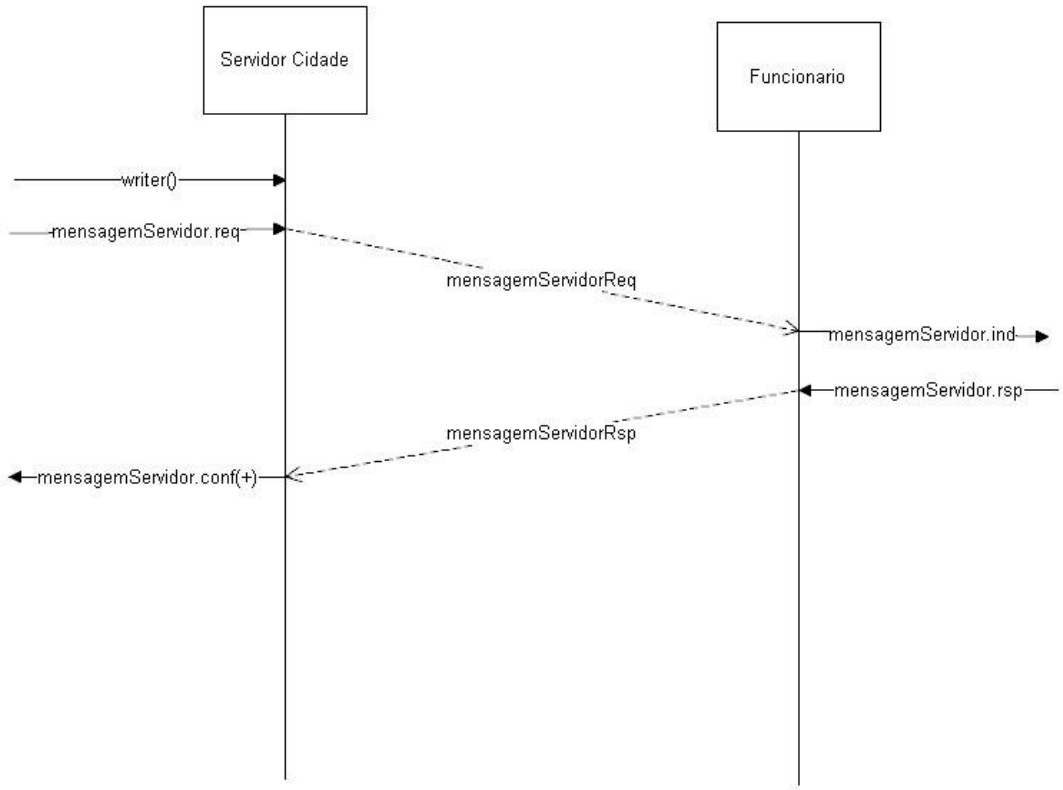
Matriculas


Servidor Cidade
Funcionario
matricula.req
matriculareq
matricula.ind
writer()
matricula.rsp
matricula
matricula.conf(+)

Mensagem Funcionário – Servidor



Mensagem Servidor – Funcionário



Primitivas de serviço

login.req – representa a ligação do Servidor Sede para o Servidor Cidade.

login.rsp – representa a resposta do Funcionário para o Servidor Cidade.

login.cnf(+) – representa a confirmação do longin.req.

ordem.req – representa a ligação do Funcionário para o Servidor Cidade.

ordem.cnf(+) - representa a confirmação do ordem.req.

ordem.rsp - representa a resposta do Servidor Cidade para o Funcionário.

confirmação.req – representa o pedido de permissão da ordem.

confirmação.rsp – representa a resposta do Funcionário para o Servidor Cidade.

confirmação.cnf(+) – representa a confirmação do confirmação.req.

matricula.req - representa a ligação do Servidor Cidade para o Funcionário.

matricula.cnf(+) - representa a confirmação do matricula.rsp.

matricula.rsp - representa a resposta do Funcionário para o Servidor Cidade.

mensagem.req - representa a ligação do Servidor Cidade para o Funcionário.

mensagem.rsp - representa a resposta do Funcionário para o Servidor Cidade.

mensagem.cnf(+) - representa a confirmação do mensagem.rsp.

mensagemServidor.req - representa a ligação do Servidor Cidade para o Funcionário.

mensagemServidor.rst – representa a resposta do F Funcionário para o Servidor Cidade.

mensagemServidor.cnf(+) - representa a confirmação do mensagemServidor.rsp.

Tipos de Dados

Formato das Primitivas

IP Origem	IP Destino	Porta Destino	Dados

- IP Origem – Contém o IP da máquina que está a enviar.
- IP Destino – Contém o IP da máquina que vai receber os dados.
- Porta Destino – Contém a porta da máquina que vai receber os dados.
- Dados – contém os dados correspondentes e específicos de cada primitiva.

Análise das limitações do programa

O nosso trabalho prático apresenta unicamente o funcionamento base que foi pedido pelo docente. Desta forma a nossa aplicação apresenta algumas limitações.

A principal limitação da aplicação é a ausência do servidor das Finanças. Sem este servidor não foi possível realizar o ponto “Validar pagamento do imposto de circulação”. A outra limitação é no ponto “Ler comunicações internas” que não está a ter o funcionamento desejado.

Desta forma o grupo reconhece que a ausência das funcionalidades avançadas é uma menos valia para o trabalho prático.

Conclusão

Através do desenvolvimento desta aplicação foram aprofundados de uma forma mais precisa os conceitos aprendidos nas aulas da disciplina. Foram bem assimilados os conceitos de comunicação existentes num sistema de cliente/servidor.

Na elaboração da aplicação a maior dificuldade sentida foi a pouco prática de programação através de protocolos de comunicação, mas que com a ajuda dos laboratórios e da ajuda dos docentes foram ultrapassadas.

Podemos concluir que a realização deste projecto foi muito benéfica para o grupo devido à aprendizagem dos conceitos importantes implícitos no mesmo que serão bastante úteis no futuro.

Anexo

Código Fonte da aplicação

Servidor Sede

- **Classe Programa**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.Threading;

namespace parquessadoSede
{
    class Program
    {
        static void Main(string[] args)
        {
            Socket socket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);

            IPAddress ip = IPAddress.Any;

            IPEndPoint ie = new IPEndPoint(ip, 6100);

            socket.Bind(ie);

            socket.Listen(10);
                    Console.WriteLine("..:: ParquesSado ::..");
            Console.WriteLine("Em espera...");


            while (true){

            Socket newSocket = socket.Accept();

            byte[] data = new byte[1024];
            string hello = "Bem vindo cidade!";
            data = Encoding.ASCII.GetBytes(hello);

            newSocket.Send(data);

            //Receber nickname

            byte[] data2 = new byte[1024];
            string nickname;
            int r_nickname;

            r_nickname = newSocket.Receive(data2);
            nickname = Encoding.ASCII.GetString(data2, 0, r_nickname);
            Console.WriteLine("A cidade " + nickname + " ligou-se a sede" );

            TrataCliente trata = new TrataCliente(newSocket, nickname);

            Thread newthread = new Thread(new ThreadStart (trata.trataligacao));
```

```
            newthread.Start();

            }
        }
    }
}
```

- **Classe TrataCliente**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.IO;



namespace parquessadoSede
{
    public class TrataCliente
    {
        Socket newSocket;
        IPEndPoint ie2;
        static Socket[]  lSocket = new Socket[20];
        static int indice = 0;
        static int ligacoes=0;
        private string mensagem;
             private string users;
             private string city;

        public TrataCliente(Socket s, string t)
        {
            newSocket = s;
            lSocket[indice++] = s;
            this.city=t;
        }

        public void trataligacao()
        {

            ie2 = (IPEndPoint)newSocket.RemoteEndPoint;
            Console.WriteLine("Cliente " + ie2.Address.ToString() + "connected");
            Console.WriteLine("Porto " + ie2.Port.ToString());
            ligacoes++;


while(true){

            //Enviar hello


                    byte[] data1 = new byte[1024];
            byte[] msg = new byte[1024];
                    int recv=0, iUsr=0;
                    //os utilizadores da aplicação
                     Funcionario[]nomes= new Funcionario[7];
            nomes[0]=new Funcionario("zona 1", "samu", "cidade 1");
```

```
            nomes[1]=new Funcionario("zona 1", "sha", "cidade 1");
            nomes[2]=new Funcionario("zona 2", "timoteo", "cidade 1");
            nomes[3]=new Funcionario("zona 2", "brunob", "cidade 1");
            nomes[4]=new Funcionario("zona 3", "eva", "cidade 1");
            nomes[5]=new Funcionario("zona 1", "trindade", "cidade 2");
            nomes[6]=new Funcionario("zona 1", "paulo", "cidade 2");
                    do{
            data1 = new byte[1024];
            msg = new byte[1024];
            recv=0;
            recv=newSocket.Receive(data1);
            users=Encoding.ASCII.GetString(data1, 0, recv);
            mensagem="nao";
            for(int i=0; i < nomes.Length; i++){

       if(users.Equals(nomes[i].getNome())&&this.city.Equals(nomes[i].getCidade()
))
            {
              mensagem="sim";
              iUsr=i;
              break;
            }

            }
            msg = new byte[1024];
                msg = Encoding.ASCII.GetBytes(mensagem);

                      newSocket.Send(msg);

                    }while(mensagem.Equals("nao"));

            newSocket.Receive(msg);
                    msg = new byte[1024];

                    msg = Encoding.ASCII.GetBytes(nomes[iUsr].getZona());

                      newSocket.Send(msg);


                      Console.WriteLine("Funcionario "+users+" conectado");
}

        }
    }
}
```

- **Classe Funcionário**

```
using System;

namespace parquessadoSede
{
       public class Funcionario
       {
              private string nome;
              private string cidade;
              private string zona;
              public Funcionario(string zona, string nome, string cidade)
              {
```

```
                this.zona=zona;
                this.nome=nome;
                this.cidade=cidade;
            }

            public string getNome(){
            return this.nome;
            }

            public string getCidade(){
            return this.cidade;
            }

            public string getZona(){
            return this.zona;
            }

            public void setZona(string s){
            this.zona=s;
            }

        }
}
```

Servidor Cidade

- **Classe Programa**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.Threading;

namespace parquessadoCidade1
{
    class Program
    {


        static void Main(string[] args)
        {

        byte[] data1;
        int recv;
        string hello;

                    Socket servidor;
            Socket funcionarios = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
                    IPEndPoint ieser = new
IPEndPoint(IPAddress.Parse("127.0.0.1"), 6100);

                    servidor = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);

            try
```

```
        {
            servidor.Connect(ieser);
        }
        catch (SocketException e)
        {
            Console.WriteLine(e.ToString());
            Console.ReadKey();
            return;
        }

                data1 = new byte[1024];
                recv = servidor.Receive(data1);
          hello = Encoding.ASCII.GetString(data1, 0, recv);
          Console.WriteLine(hello);




          IPAddress ip = IPAddress.Any;

        IPEndPoint ie = new IPEndPoint(ip, 6000);


        funcionarios.Bind(ie);

        funcionarios.Listen(10);
        Console.WriteLine("..:: ParquesSado ::..");
        Console.WriteLine("A espera...");



        byte[] data2;

                       data2 = new byte[1024];


                       data2 = Encoding.ASCII.GetBytes("cidade 1");

                    servidor.Send(data2);



        while (true){

        Socket aceptFuncionario = funcionarios.Accept();

        TrataCliente trata = new TrataCliente(aceptFuncionario, servidor);

        Thread newthread = new Thread(new ThreadStart (trata.trataligacao));

        Thread writer= new Thread(new ThreadStart(trata.writer));


            newthread.Start();
            writer.Start();


        }
    }
}
```

```
}
```

- **Classe TrataCliente**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.IO;


namespace parquessadoCidade1
{
    public class TrataCliente
    {

        private Socket servidor;


        Socket funcionario;
        IPEndPoint ie2;
        static Socket[]  lSocket = new Socket[20];
        static int indice = 0;
        static int ligacoes=0;
        private byte[] data;
        private string hello;
        private int r_nickname;
        private string nickname;
        private string answer;
        private string zona;
        private int recv;
        private string mensagem;
        private Random rand = new Random();
        int rdm;
        public TrataCliente(Socket s, Socket t)
        {
            funcionario = s;
            lSocket[indice++] = s;
            servidor=t;
        }

        public void trataligacao()
        {
             data=new byte[1024];
        hello=null;
        r_nickname=0;
        nickname=null;
        answer=null;
        recv=0;
        mensagem=null;
              //as matriculas da cidade
              Matricula [] matriculas= new Matricula[8];
              matriculas[0]=new Matricula("zona 1", "12-56-mq");
              matriculas[1]=new Matricula("zona 1", "30-31-sm");
              matriculas[2]=new Matricula("zona 2", "39-33-aa");
              matriculas[3]=new Matricula("zona 2", "90-19-zz");
              matriculas[4]=new Matricula("zona 3", "01-05-mm");
```

```
                matriculas[5]=new Matricula("zona 3", "17-15-rr");
                matriculas[6]=new Matricula("zona 1", "08-05-rm");
                matriculas[7]=new Matricula("zona 1", "11-50-gd");

               ie2 = (IPEndPoint)funcionario.RemoteEndPoint;
               Console.WriteLine("Cliente " + ie2.Address.ToString() + "connected");
               Console.WriteLine("Porto " + ie2.Port.ToString());
               ligacoes++;


              data = new byte[1024];
               hello = "Bem vindo funcionario!";
               data = Encoding.ASCII.GetBytes(hello);

               funcionario.Send(data);

               //Receber nickname

               do{
                 data = new byte[1024];
               r_nickname = funcionario.Receive(data);
               nickname = Encoding.ASCII.GetString(data, 0, r_nickname);
               Console.WriteLine("O funcionario " + nickname + " esta a tentar
ligar-se a cidade " + ligacoes + "." );

               data = new byte[1024];


                     data = Encoding.ASCII.GetBytes(nickname);

               servidor.Send(data);

               data = new byte[1024];
               r_nickname=0;

                       r_nickname = servidor.Receive(data);
                 answer = Encoding.ASCII.GetString(data, 0, r_nickname);

                 data=Encoding.ASCII.GetBytes(answer);
                 funcionario.Send(data);

                 Console.WriteLine(answer);

         }

                 while(answer.Equals("nao"));

               //Enviar nome da cidade a sede


               data = new byte[1024];
               zona=null;
               servidor.Send(data);
               r_nickname=0;
                       data = new byte[1024];
                       r_nickname = servidor.Receive(data);
                 zona = Encoding.ASCII.GetString(data, 0, r_nickname);

                 mensagem=null;
                 data = new byte[1024];
               Boolean pertence=false;
               do
```

```
            {
                recv=0;
              recv = funcionario.Receive(data);
                hello = Encoding.ASCII.GetString(data, 0, recv);


                if(hello.Equals("ordens")){
                    Console.WriteLine(nickname + " esta a consultar as ordens");
                    rdm=rand.Next(3)+1;
                    mensagem="Cidade 1 > Vai para a zona "+rdm;

                data = Encoding.ASCII.GetBytes(mensagem);

                funcionario.Send(data);

                }

                else if(hello.Length==8 && hello[2]==45 && hello[5]==45){

                    Console.WriteLine(nickname + " verifica se a matricula
"+hello+" e valida");

             for(int j=0; j<matriculas.Length; j++){


       if(hello.Equals(matriculas[j].getMatricula())){pertence=true;}

             }

                    if(pertence==true){

                    mensagem=hello + " > matricula valida";

                data = Encoding.ASCII.GetBytes(mensagem);

                funcionario.Send(data);

                    }
                    else{

                           mensagem=hello + " > matricula invalida - Multa 50
euros";

                data = Encoding.ASCII.GetBytes(mensagem);

                funcionario.Send(data);

                    }
                }

                else{
                    Console.WriteLine(nickname +" > "+hello);
                mensagem=nickname + "> " + hello;

                data = Encoding.ASCII.GetBytes(mensagem);

                enviaMsgFunc(data);

                }

            } while (hello.CompareTo("exit") != 0);
```

```
            ligacoes--;
            Console.WriteLine("Cliente " + ie2.Address.ToString() +
"Desconectado");
            Console.WriteLine("O utilizador " + nickname + " desligou a sessão e
ficaram " + ligacoes + " utilizador(es).");
            funcionario.Close();



            Console.ReadLine();

        }
        //para o servidor da cidade enviar informacoes para todos os funcionarios
                public void writer()
        {


            byte[] data = new byte[1024];
            string hello2;
            byte[] data2 = new byte[1024];

            do
            {
                hello2 = Console.ReadLine();
                data = Encoding.ASCII.GetBytes("Cidade 1 > "+hello2);
                for(int i=0; i< ligacoes; i++){
                    if(lSocket[i].Connected){
                           lSocket[i].Send(data);
                    }
                    //cliente.Send(data);
                }
            } while (hello2.CompareTo("exit") != 0);

            Console.ReadKey();




        }

                public void enviaMsgFunc(byte[]ms)
                {
                                // Broadcast
                for (int i = 0; i < indice; i++)

                {
                    if (lSocket[i].Connected){

                        lSocket[i].Send(ms);
                    }
                }
                }


}
}
```

- **Classe Matricula**

```
using System;


namespace parquessadoCidade1
{

	public class Matricula
	{
		private string zona;
		private string matricula;
		public Matricula(string zona, string matricula)
		{
			this.zona=zona;
			this.matricula=matricula;
		}

		public string getZona(){
		return this.zona;
		}

		public string getMatricula(){
		return this.matricula;
		}

	}
}
```

Funcionário

- **Classe Programa**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net.Sockets;
using System.Net;
using System.Threading;

namespace parquessadoFuncionario1
{
	  class Program
    {
       static void Main(string[] args)
        {

			ThreadCom tc= new ThreadCom();
			tc.comunicar();

       }
       }

    }
```

- **Classe ThreadCom**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net.Sockets;
using System.Net;
using System.Threading;

namespace parquessadoFuncionario1{
       public class ThreadCom
       {

             private Thread tReader;
        private Thread tWriter;

       private string nicknameT;
       private string nickname;

        private byte[] data1;
        private int recv;
        private string hello;

        private Socket cliente;



             public ThreadCom()
             {

             }



        public void comunicar()
        {
             IPEndPoint ie = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 6000);

             cliente = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);

            try
            {
                cliente.Connect(ie);
            }
            catch (SocketException e)
            {
                Console.WriteLine(e.ToString());
                Console.ReadKey();
                return;
            }



                    data1 = new byte[1024];
                    recv = cliente.Receive(data1);
             hello = Encoding.ASCII.GetString(data1, 0, recv);
                    Console.WriteLine(hello);



            byte[] data = new byte[1024];
```

```
        string answer;
        byte[] data2 = new byte[1024];
        int int_answer;
        do{

        Console.WriteLine("Insira o seu username");
        nicknameT = Console.ReadLine();
        Console.WriteLine("O seu username é " + nicknameT + ".");
        data2 = Encoding.ASCII.GetBytes(nicknameT);
        cliente.Send(data2);
        int_answer=0;
        data = new byte[1024];
        int_answer = cliente.Receive(data);
        answer = Encoding.ASCII.GetString(data, 0 , int_answer);
        Console.WriteLine(answer);
        if (answer.Equals("nao")){
        Console.WriteLine("Username errado, tente novo");
        }
        }while(answer.Equals("nao"));

          tReader = new Thread(new ThreadStart(reader));
          tWriter = new Thread(new ThreadStart(writer));



                tWriter.Start();
        tReader.Start();

     }

     public void reader()
    {

        byte[] data1 = new byte[1024];
        int recv;
        string hello;


        do
        {
            recv = cliente.Receive(data1);
            hello = Encoding.ASCII.GetString(data1, 0, recv);

            Console.WriteLine(hello);
        } while (true);

     }

     public void writer()
     {


        byte[] data = new byte[1024];
        string hello2;
        byte[] data2 = new byte[1024];

        do
        {
            hello2 = Console.ReadLine();
            data = Encoding.ASCII.GetBytes(hello2);
            cliente.Send(data);
```

```
            } while (hello2.CompareTo("exit") != 0);

            tReader.Abort();
            cliente.Shutdown(SocketShutdown.Both);
            cliente.Close();

            Console.ReadKey();



        }


    }
}
```