

Tópicos de Segurança

# Protocolo SI Threads

Cofinanciado por:



UNIÃO EUROPEIA  
Fundo Social Europeu

# Protocolo SI

Com a utilização da biblioteca **ProtocolSI**, desenvolvida pelo DEI, a troca de informação pela rede de dados através do protocolo TCP torna-se mais simples de implementar.

# Protocolo SI

Após observar a documentação da classe **ProtocolSI**, presente nesta biblioteca, retiram-se alguns métodos e propriedades relevantes:

- **Make()**: (cria uma mensagem/pacote de um tipo específico)
- **GetData()**: (obtém a última mensagem recebida)
  - Variações: **GetIntFromData()**; **GetStringFromData()**;
- **GetCmdType()**: (interpreta o tipo de mensagem/pacote recebido)
- **Buffer**: (propriedade que permite armazenar a mensagem/pacote recebida)

# Protocolo SI

Nesta biblioteca também é possível observar uma enumeração:

## **ProtocolSICmdType.**

Esta enumeração permite indicar o tipo de mensagem que poderá estar a ser enviada. Em seguida são apresentados alguns exemplos:

- ACK
- DATA
- SYM\_CIPHER\_DATA
- SECRET\_KEY
- IV
- EOF (End Of File)
- EOT (End Of Transmission)
- PUBLIC\_KEY

# Protocolo SI

Exemplos de utilização:

## Enviar:

```
string msg = "TopSeg";  
byte[] packet = protocolSI.Make(ProtocolSICmdType.DATA, msg);  
networkStream.Write(packet, 0, packet.Length);
```

## Receber:

```
networkStream.Read(protocolSI.Buffer, 0, protocolSI.Buffer.Length);  
Console.WriteLine(protocolSI.GetStringFromData());
```

# Threads

**Threads** são unidades de execução dentro de um processo; um conjunto de instruções.

Por omissão, um programa em C# apenas tem uma **thread**, conhecida como **thread principal**. No entanto, threads auxiliares podem ser criadas e usadas para executar código em paralelo com a thread principal.

# Threads

Podem-se utilizar múltiplas **threads** para monitorizar o input do utilizador, executar tarefas em background e para tratar de vários inputs em simultâneo.

Estas partilham os recursos da aplicação.

De forma a permitir a programação dessas múltiplas **threads**, a Framework .NET implementa várias classes e interfaces no *namespace* **System.Threading**.


# Threads

Através de um pequeno exemplo em C#, é possível visualizar e compreender como as **threads** funcionam.

```
static void Main(string[] args)
{
    Thread threadLenta = new Thread(slowThreadHandler);
    Thread threadRapida = new Thread(fastThreadHandler);
    threadLenta.Start();
    threadRapida.Start();
}

static private void slowThreadHandler()
{
    while (true)
    {
        Console.WriteLine("Olá! Sou a thread lenta...");
        Thread.Sleep(3000);
    }
}

static private void fastThreadHandler()
{
    while (true)
    {
        Console.WriteLine("Olá! Sou a thread rápida...");
        Thread.Sleep(1000);
    }
}
```



```
Olá! Sou a thread lenta...
Olá! Sou a thread rápida...
Olá! Sou a thread rápida...
Olá! Sou a thread rápida...
Olá! Sou a thread lenta...
Olá! Sou a thread rápida...
Olá! Sou a thread rápida...
Olá! Sou a thread rápida...
Olá! Sou a thread lenta...
Olá! Sou a thread rápida...
Olá! Sou a thread rápida...
```



# Ficha nº3

Cofinanciado por:



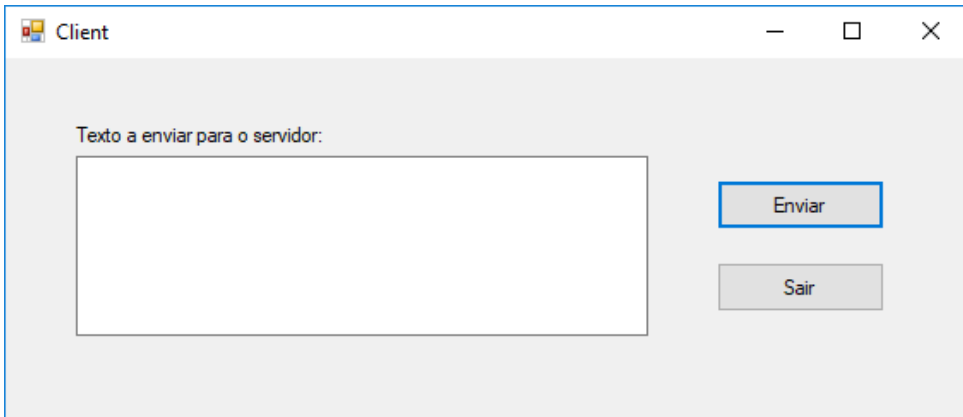
UNIÃO EUROPEIA  
Fundo Social Europeu

## Ficha 3

O próximo exercício pretende mostrar as utilidades da biblioteca **ProtocolSI**, assim como a utilização de **threads**, implementados em .NET, de forma a permitir a ligação de vários clientes a um servidor.

1. Elabore um cenário Cliente-Servidor, em *Windows Forms Application* e em *Console Application*, respetivamente, com o objetivo de criar uma aplicação que apresente frases enviadas pelo(s) cliente(s).

# Ficha 3



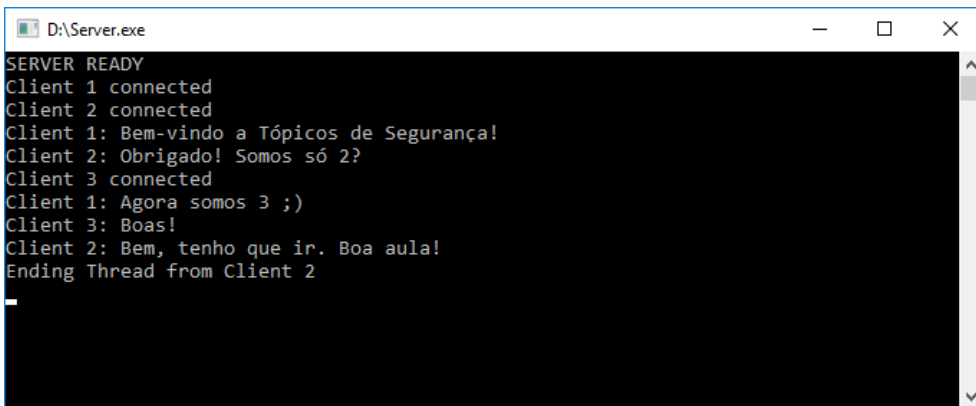
Client

Texto a enviar para o servidor:

Enviar

Sair

Formulário de um cliente



D:\Server.exe

```
SERVER READY
Client 1 connected
Client 2 connected
Client 1: Bem-vindo a Tópicos de Segurança!
Client 2: Obrigado! Somos só 2?
Client 3 connected
Client 1: Agora somos 3 ;)
Client 3: Boas!
Client 2: Bem, tenho que ir. Boa aula!
Ending Thread from Client 2
```

Formulário do servidor

# Ficha 3

## Requisitos

- Servidor:
  - Deverá permitir a ligação de múltiplos clientes.
- Cliente(s):
  - Deverá enviar um EOT no final da transmissão.
- Ambos:
  - Deverão utilizar a biblioteca **ProtocolSI** para comunicar.