

Trabalho Prático em Grupo

Tópicos abordados:

- Sockets TCP/IP em .NET
- Algoritmos criptográficos em .NET
- Autenticação

Este trabalho prático é para ser elaborado por grupos de **dois a três estudantes**.

Este trabalho engloba duas **provas orais individuais**, denominado defesa que será elaborado no ato da entrega da fase I e fase final do trabalho prático.

Estão previstas **aulas de apoio ao trabalho prático**.

1. Objetivos

O objetivo deste projeto é o desenvolvimento de um chat com troca de mensagens e ficheiros de forma segura, em C#. O trabalho será composto por um módulo cliente e por um módulo servidor, com as seguintes características base:

a) **O cliente**, com User Interface (UI), pode:

- Enviar a sua chave pública;
- Autenticar-se no servidor fornecendo as credenciais;
- Enviar e receber as mensagens de conversação e ficheiros;
- Tornar todas as comunicações o mais seguras possível;
- Validar todas as mensagens e ficheiros trocados com recurso a assinaturas digitais.

b) **O servidor**, sem UI, permite:

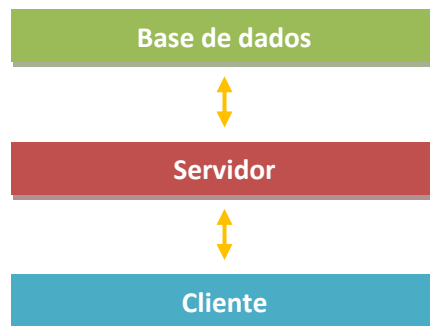
- Receber ligações de cliente;
- Guardar a chave pública do cliente;
- Autenticar um utilizador já registado no sistema;
- Validar as assinaturas do cliente;
- Enviar e receber as mensagens de conversação e ficheiros partilhados de forma segura;
- Receber e processar os dados relativos às mensagens e ficheiros partilhados de forma segura.

2. Interface gráfica

O **módulo de cliente** deverá permitir tudo o que está estipulado no ponto anterior, com a possibilidade de cada grupo apresentar um UI à escolha.

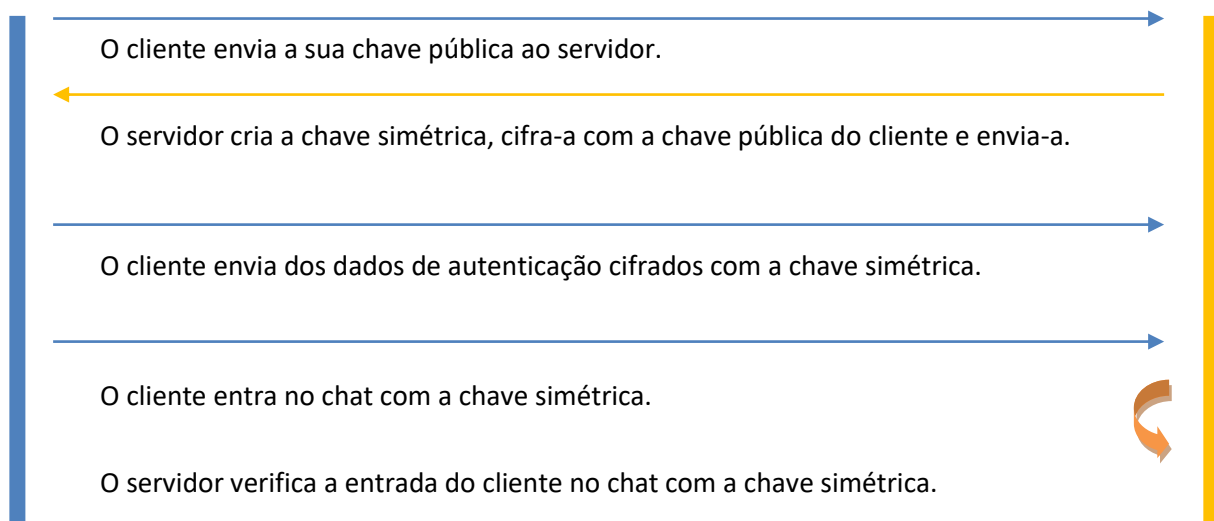
3. Esquema

O esquema seguinte apresenta o fluxo principal do sistema a desenvolver:



4. Comunicação

A figura seguinte exemplifica parte da comunicação existente entre o cliente e o servidor, nomeadamente na parte da autenticação e no acesso ao chat de mensagens.



Notas:

- A utilização da biblioteca fornecida (**ProtocolSI.dll**) é obrigatória;
- No armazenamento das credenciais deverá ser utilizado um *salt* aleatório para cada utilizador;
- **Um servidor aceita multi-clientes (época de exame)**;
- O **código deverá ser comentado** e todas as **funções terão de ter uma explicação** sobre a sua **funcionalidade e objetivos**.
- Em caso de dúvida sobre algum ponto neste enunciado **deverá contactar sempre o docente**.
- **Links úteis:**
 - <https://docs.microsoft.com/pt-br/dotnet/api/system.collections.generic.list-1?view=netframework-4.8>
 - <http://snippetbank.blogspot.com/2014/04/csharp-client-server-broadcast-example-1.html>

Épocas de Avaliação**a. Avaliação Contínua**

i. Apresenta duas fases de entrega do projeto:

1. **Fase I** – corresponde a **30% da nota final do projeto**. É pedido ao grupo que realize uma prova oral individual dos seguintes elementos, implementados no projeto:
 - Relatório de Análise de Requisitos.
 - Desenvolvimento do User Interface:
 - i. Aplicação “*Windows Form*” para o cliente
 - ii. Aplicação “*Console*” para o servidor
 - Não é necessário implementar cifragem nas mensagens trocadas para esta etapa.
2. **Fase II** – corresponde a **70% da nota final do projeto**. É pedido ao grupo que realize uma prova oral individual dos seguintes elementos, implementados no projeto:
 - Relatório de Análise de Requisitos com todos os requisitos desenvolvidos.
 - Desenvolvimento e implementação de código para:
 - i. Chat a funcionar entre clientes e servidor em que o **servidor suporta um cliente**, permitindo o envio de **mensagens e ficheiros cifrados** e devidamente guardados.
 - Criação de um log (.txt) do sistema para guardar todos os dados processados pelo servidor.
 - User Interface final.

5. Regras de entrega do projeto final (avaliação periódica)

- A data de entrega da **Fase I** está agendada para dia **19 de abril** às **09h30**, no **moodle**, com **prova oral individual a ser agendada na plataforma Microsoft Teams ou em regime presencial**;
- A data final de entrega é a **21 de junho** às **9h30** com **prova oral individual a ser agendada na plataforma Microsoft Teams ou em regime presencial** no dia **25 de junho**;
- O projeto deve ser entregue num **ficheiro único comprimido** com a identificação no seguinte formato: **NumeroEstudante1 NumeroEstudante2 NumeroEstudante3**

Nas avaliações de época normal e recurso, o grupo de estudantes deverá implementar obrigatoriamente uma **funcionalidade extra na base do projeto da avaliação periódica**.

a. **Época Normal** - corresponde a **100% da nota final do projeto**. É pedido ao grupo que realize uma prova oral individual dos seguintes elementos, implementados no projeto:

- Relatório de Análise de Requisitos.
- User Interface desenvolvido e implementado (Aplicação “*Windows Form*” para o cliente e Aplicação “*Console*” para o servidor).
- Desenvolvimento e implementação de código para:
 - i. Chat a funcionar entre clientes e servidor em que o **servidor suporta um cliente**, permitindo o envio de **mensagens e ficheiros cifrados** e devidamente guardados.
- Criação de um log (.txt) do sistema para guardar todos os dados processados pelo servidor.
- Criação de um registo estatístico dos dados processados pelo servidor.

b. **Época Recurso** - corresponde a **100% da nota final do projeto**, é pedido ao grupo que realize uma prova oral individual dos seguintes elementos, implementados no projeto:

- Relatório de Análise de Requisitos.
- User Interface desenvolvido e implementado (Aplicação “*Windows Form*” para o cliente e Aplicação “*Console*” para o servidor).
- Desenvolvimento e implementação de código para:
 - i. Chat a funcionar entre clientes e servidor em que o **servidor suporta n cliente**, permitindo o envio de **mensagens e ficheiros cifrados** e devidamente guardados.
- Criação de um log (.txt) do sistema para guardar todos os dados processados pelo servidor.
- Criação de um registo estatístico dos dados processados pelo servidor.

6. Regras de entrega do projeto final (época de exame e recurso)

- A data final do projeto em ÉPOCA DE EXAME tem a **entrega a 25 de junho** às **9h30**, moodle, com **prova oral individual a ser agendada na plataforma Microsoft Teams ou em regime presencial nesse dia às 14h30**;
- A data final do projeto em ÉPOCA DE RECURSO tem a **entrega a 17 de julho** às **9h30** com **prova oral individual a ser agendada na plataforma Microsoft Teams ou em regime presencial nesse dia às 14h30**;
- O projeto deve ser entregue num **ficheiro único comprimido** com a identificação no seguinte formato: **NumeroEstudante1 NumeroEstudante2 NumeroEstudante3**

7. Critérios de avaliação

Critérios	Peso (%)
Utilização de Criptografia Assimétrica	15%
Utilização de Criptografia Simétrica	15%
Troca de Mensagens e Ficheiros	15%
<i>Threads</i>	15%
Autenticação	10%
Validação dos Dados	10%
Apresentação do código	10%
User Interface	5%
Lógica do Chat	5%
<i>Total</i>	100%

ANEXO**Exemplos de código em C#:**

```
//EXEMPLO PEDIDO CLIENTE
//*****
// (...)
// VARIÁVEL PARA RECEBER OS DADOS
string textAux = "";
// CRIA UMA MENSAGEM DO TIPO USER_OPTION_1 (PROTOCOLO SI), PODEM USAR OS VÁRIOS TIPOS PARA
VÁRIAS FUNÇÕES
    byte[] opt1 = protocolSI.Make(ProtocolSICmdType.USER_OPTION_1);
// ENVIA O PEDIDO PARA O SERVIDOR (WRITE)
    networkStream.Write(opt1, 0, opt1.Length);
// ENQUANTO HOUVER COISAS PARA RECEBER (OS DADOS PODEM TER SIDO DIVIDIDOS PARA SEREM
ENVIADOS)
    while (true)
    {
        // LÊ A RESPOSTA QUE CHEGOU (READ)
        networkStream.Read(protocolSI.Buffer, 0, protocolSI.Buffer.Length);
        // SE FOR O FIM DA RESPOSTA SAI FORA
        if (protocolSI.GetCmdType() == ProtocolSICmdType.EOF)
        {
            // SAI FORA DO WHILE
            break;
        }
        // SENÃO, E SE FOREM DADOS ESCRIVE PARA A STRING
        else if (protocolSI.GetCmdType() == ProtocolSICmdType.DATA)
        {
            // ESCRIVE OS DADOS PARA A STRING
            textAux = textAux + protocolSI.GetStringFromData();
        }
    }
// ATUALIZA O TEXTO DA TEXTBOX
    textbox.Text = textAux;
// (...)
```

```
//EXEMPLO RESPOSTA SERVIDOR
//*****
// (...)
// FOREVER RUNNING ALONE...
while (true)
{
    // RECEBE O PEDIDO DO CLIENTE (READ)
    int bytesRead = networkStream.Read(protocolSI.Buffer, 0, protocolSI.Buffer.Length);
    // VERIFICA O TIPO DE MENSAGEM RECEBIDO
    if (protocolSI.GetCmdType() == ProtocolSICmdType.USER_OPTION_1){
        // TEXTO QUE VAI SER ENVIADO COMO RESPOSTA
        string response = "abc1234567890";
        // LIMPA A VARIÁVEL AUXILIAR
        string stringChunk = "";
        // TAMANHO PARA LER DE CADA VEZ (USAR COMO MÁX 64 CARACTERES)
        int chunkSize = 4;
        // VAI BUSCAR O TAMANHO DA RESPOSTA
        int stringLength = response.Length;
        // PERCORRE A RESPOSTA E VAI DIVIDINDO EM PEDAÇOS PEQUENOS (CHUNKS)
        for (int i = 0; i < response.Length; i = i + chunkSize)
        {
            // CASE SEJA O ÚLTIMO CHUNK
            if (chunkSize > stringLength)
            {
                // ENVIA TUDO O QUE FALTA
                stringChunk = response.Substring(i);
            }
            // CASO SEJA UM CHUNK NORMAL
            else
            {
                // DECREMENTA O TOTAL DE CARACTERES JÁ LIDOS
                stringLength = stringLength - chunkSize;
                // OBTÉM ESSE CHUNK
                stringChunk = log.Substring(i, chunkSize);
            }
            // CRIA A MENSAGEM DO TIPO DATA UTILIZANDO O PROTOCOLO SI
            byte[] packet = protocolSI.Make(ProtocolSICmdType.DATA, stringChunk);
            // ENVIA A RESPOSTA PARA O CLIENTE (WRITE)
            networkStream.Write(packet, 0, packet.Length);
        }
        // CRIA O EOF PARA ENVIAR PARA O CLIENTE
        byte[] eof = protocolSI.Make(ProtocolSICmdType.EOF);
        // ENVIA A RESPOSTA PARA O CLIENTE (WRITE)
        networkStream.Write(eof, 0, eof.Length);
    }
}
// (...)
```

/EXEMPLO LEITURA DE FICHEIROS

```
//*****  
// (...)  
// LOCALIZAÇÃO DO TXT  
    string path = @"ficheiro.txt";  
// LÊ TODO O TEXTO DO FICHEIRO  
    string txtContent = File.ReadAllText(path);  
// (...)
```

//EXEMPLO ESCRITA DE FICHEIROS

```
//*****  
// (...)  
    string textAdd ="Mais texto!";  
// LOCALIZAÇÃO DO TXT  
    string path = @"ficheiro.txt";  
    // LIMPA O TXT E ESCRIVE A STRING COM UMA NOVA LINHA NO FIM, COM CODIFICAÇÃO UTF8  
File.WriteAllText(path, textAdd + Environment.NewLine, Encoding.UTF8);  
// ADICIONA AO TXT A STRING COM UMA NOVA LINHA NO FIM, COM CODIFICAÇÃO UTF8  
    File.AppendAllText(path, textAdd + Environment.NewLine, Encoding.UTF8);  
// (...)
```