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 uma **prova oral e um teste-prático**, que serão elaborados no ato da entrega da fase I e fase final do trabalho prático, respetivamente.

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

1. Objetivos

O objetivo deste projeto é o desenvolvimento de um chat com troca de mensagens de forma segura, em C# (*Windows Forms*, *Console Application* e *Web App*). O trabalho será composto por módulo cliente e por módulo servidor, com as seguintes características base:

a) **Módulo 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;
- Tornar todas as comunicações o mais seguras possível;
- Validar todas as mensagens trocados com recurso a assinaturas digitais.

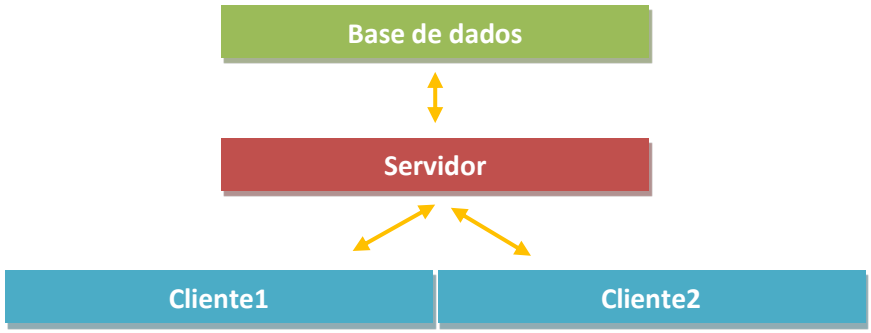
b) **Módulo servidor**, sem UI, permite:
- Receber ligações de cliente;
- Guardar a chave pública do cliente;
- Autenticar um utilizador já registado no sistema;
- Validar as assinaturas do cliente;
- Enviar e receber as mensagens de conversação de forma segura;
- Receber e processar os dados relativos às mensagens 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.

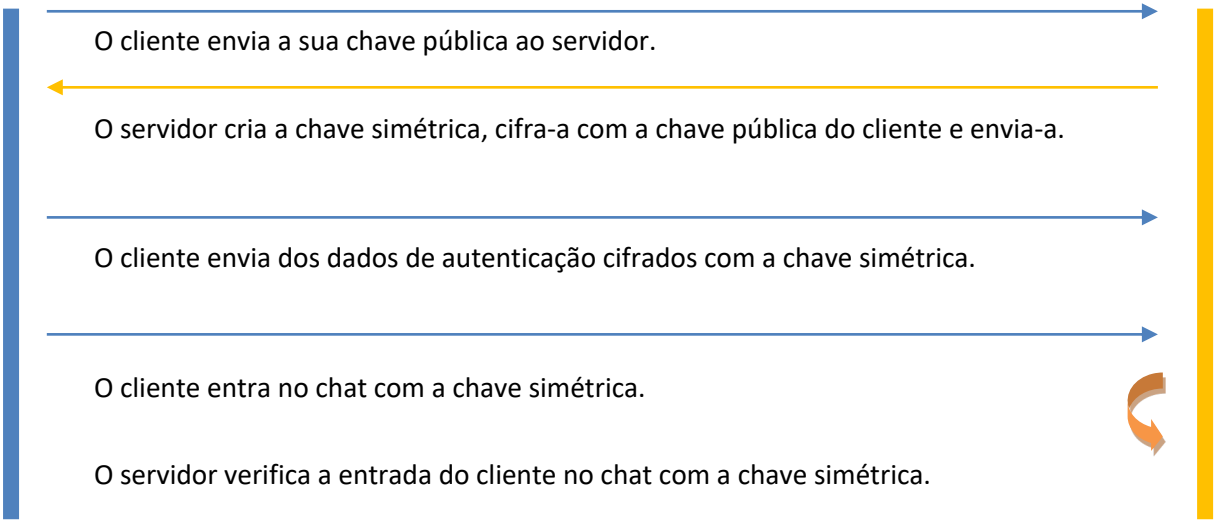
3. Esquema

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 multiclientes (época de exame e recurso)**;
- 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

Cofinanciado por:

Épocas de Avaliação

AVALIAÇÃO CONTÍNUA

i. Apresenta duas fases de entrega do projeto:
   1. **Fase I** – corresponde a **20% da nota final do projeto**. É pedido ao grupo que realize uma prova oral dos seguintes elementos, implementados no projeto:
      - Relatório de Análise de Requisitos
      - Desenvolvimento *Solution*:
        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 **80% da nota final do projeto**. Com os seguintes elementos, implementados no projeto:
      - Desenvolvimento e implementação de código para:
        i. Chat a funcionar entre os dois clientes e servidor em que o **servidor suporta os dois clientes**, permitindo o envio de **mensagens cifradas** e devidamente guardados.
      - Criação de um log (.txt) do sistema para guardar todos os dados processados pelo servidor.
      - User Interface final.

Regras de entrega do projeto final (avaliação periódica)

- A data de entrega da **Fase I** está agendada para dia **22 de abril** até às **23h59, no moodle**, com **prova oral individual**, no dia **24 de abril**, em aula.
- A data final de **entrega é a 13 de junho** até às **23h59**, no moodle e com **um teste-prático correspondente a 100% da nota final** no dia **27 de junho**;
- O projeto deve ser entregue num **ficheiro único comprimido** com a identificação no seguinte formato: **NumeroEstudante1_NumeroEstudante2_NumeroEstudante3**.

ÉPOCA NORMAL E RECURSO

O grupo de estudantes deverá implementar obrigatoriamente uma **funcionalidade extra na base do projeto da avaliação periódica**.

Época Normal - corresponde a **100% da nota final do projeto**. Com os seguintes elementos, implementados no projeto:

- Relatório de Análise de Requisitos.
- User Interface desenvolvido e implementado (Aplicação *Windows Form* para os diversos clientes 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 os diversos clientes**, 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.
- Criação de um módulo web com autenticação e autorização utilizando a *framework* ASP.NET permitindo, após a autenticação, o download do ficheiro log.

Época Recurso - corresponde a **100% da nota final do projeto**, com os 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 clientes**, 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.
- Criação de um módulo web com autenticação e autorização utilizando a *framework* ASP.NET permitindo, após a autenticação, o download do ficheiro log.

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

- A data final do projeto em ÉPOCA DE EXAME com **entrega a 27 de junho, no moodle,** e um **teste-prático correspondente a 100% da nota final**.
- A data final do projeto em ÉPOCA DE RECURSO com **entrega a 10 de julho, no moodle,** e um **teste-prático correspondente a 100% da nota final**.
- O projeto deve ser entregue num **ficheiro único comprimido** com a identificação no seguinte formato: **NumeroEstudante1_NumeroEstudante2_NumeroEstudante3**

5. Critérios de avaliação

Critérios	Peso (%)
Utilização de Criptografia Assimétrica	15%
Utilização de Criptografia Simétrica	15%
Troca de Mensagens (Ficheiros – época de exames)	15%
Threads	15%
Autenticação	10%
Validação dos Dados	10%
Apresentação do código	10%
User Interface	5%
Lógica do Chat	2,5%
Extra	*2,5%*
Total	100%

ANEXO

Exemplos de código em C#:

```
//EXEMPLO PEDIDO CLIENTE
//*****************************************************************
// (...)
// VARIAVEL 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 ESCREVE PARA A STRING
                else if (protocolSI.GetCmdType() == ProtocolSICmdType.DATA)
                {
                        // ESCREVE OS DADOS PARA A STRING
                        textAux = textAux + protocolSI.GetStringFromData();
                }
        }
// ATUALIZA O TEXTO DA TEXTBOX
        textbox.Text = textAux;
// (...)
```

```
//EXEMPLO RESPOSTA SERVIDOR
/
/
/
w
{
```

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

```
            if (protocolSI.GetCmdType() == ProtocolSICmdType.USER_OPTION_1){
```

```
//EXEMPLO ESCRITA DE FICHEIROS
//*****************************************************************
// (...)
        string textAdd ="Mais texto!";
// LOCALIZAÇÃO DO TXT
        string path = @"ficheiro.txt";
        // LIMPA O TXT E ESCREVE 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);
// (...)
```

```
                        if (chunkSize > stringLength)
                        {
                        // ENVIA TUDO O QUE FALTA
                                stringChunk = response.Substring(i);
                        }
                        // CASO SEJA UM CHUNK NORMAL
                        else
                        {
                                // DECREMENTA O TOTAL DE CHARACTERES JÁ LIDOS
                                        stringLength = stringLength - chunkSize;
                                // OBTEM 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);
        }
}
// (...)
```