

**Relatório de Segurança e Confiabilidade**

Trabalho 1

Engenharia Informática

Grupo 51

João David n49448

João Marques n49038

Luís Moreira n49531

# Objectivos concretizados

# Organização do Software

Na implementação do projecto, foram criados dois programas, o MsgFile (cliente) que possui uma linha de comandos que permite comunicar com o outro programa, MsgFileServer. Para estabelecer um protocolo de comunicação, foi criada uma class Message que “carrega” o conteúdo das mensagens pela rede (explicada em maior detalhe mais à frente). De forma a organizar todas as classes, foram criados três packages, o package ‘client’ que contem as classes que implementam o MsgFile (inclusive), o package ‘server’ que contem todas as classes que implementam o MsgFileServer(inclusive), nestes dois packages, as classes contidas não “conhecem” outras classes fora do seu package, excepto as classes do terceiro package, ‘communication’, que contém a class Message referida anteriormente e o enumerado OpCode que define os códigos das operações.

Package client

* Client
* ClientConst
* MsgFile

O MsgFile corresponde ao programa cliente, ao ser passado como argumento o nome e password do utilizador é criado um Client que é responsável por comunicar com o servidor, após o utilizador se ter conectado com sucesso é disponibilizado a linha de comandos, o parsing é feito pelo MsgFile . No ClientConst temos os caminhos dos directórios definidos para guardar os ficheiros transferidos e mensagens colectadas do servidor.

Package communication

* Message
* OpCode

Package server

* ConcurrentManager
* Manager
* MsgFileServer
* ServerConst
* Skel

# Formato das mensagens

Para concretizar o envio de mensagens foi criada uma class objecto ‘Message’ com diversos atributos que são usados dependendo do propósito da mensagem, para definir que conteúdo a mensagem contêm efectivamente foi criado um enum ‘OpCode’, sempre que uma mensagem é criada é definido um OpCode para que o receptor da mensagem saiba que atributos da mensagem foram inicializados, para que os possa aceder e concretizar a operação requerida, em certos casos a mensagem apenas contêm o OpCode inicializado visto não ser necessário mais informação para que seja feita a operação, por exemplo a operação “list” vinda do cliente para o servidor apenas leva o opCode com ‘LIST\_FILES’.

O OpCode também tem como função na resposta do servidor para o cliente definir se as operações foram bem concretizadas, ou em caso de erro informar o cliente o que aconteceu, por exemplo, ERR\_NOT\_REGISTERED, quando o utilizador introduzido como input numa operação não esteja registado no servidor.

Atributos de Message

* OpCode opCode
* OpCode[] arrCode
* String str
* String[] arrStr
* ArrayList<String> arrListStr
* ArrayList<Byte[]> arrListArrByte
* Byte[] arrByte

Operation Codes (OpCode)

* STORE\_FILES, LIST\_FILES, REMOVE\_FILES, USERS, TRUST\_USERS, UNTRUST\_USERS, DOWNLOAD\_FILE, SEND\_MSG, COLLECT\_MSG, END\_CONNECTION
* OP\_SUCCESSFUL: operação sobre no servidor bem-sucedida
* OP\_RES\_ARRAY: os resultados da operação vêm num array para cada parâmetro enviado para o servidor respectivamente, ex:”trusted fernando carlos rita” retorna um vetor de OpCode com length 3
* OP\_ERROR: operação no servidor não foi bem-sucedida, exception por exemplo
* ERR\_NOT\_REGISTERED: user enviado como parâmetro não está registado no servidor
* ERR\_ALREADY\_EXISTS: parâmetro introduzido já existe, por exemplo fazer “trusted fernando”, quando o fernando já tinha sido trusted anteriormente
* ERR\_NOT\_FRIENDS: o utilizador local e o enviado como parâmetro não são amigos
* ERR\_YOURSELF: utilizador enviado como parâmetro é o utilizador local
* ERR\_NOT\_FOUND: parâmetro enviado não encontrado, por exemplo, ao fazer download de um ficheiro inexistente, ou fazer untrusted de um utilizador que não foi trusted anteriormente

|  |  |  |
| --- | --- | --- |
| **Comando Utilizador** | **Mensagem de pedido** | **Mensagem de resposta** |
| store <files> | STORE\_FILES <arrListStr> <arrListArrBytes> | OP\_RES\_ARRAY <arrCode> |
|  |  | OP\_ERROR |
| list | LIST\_FILES | OP\_SUCCESSFUL <arrStr> |
|  |  | OP\_ERROR |
| remove <files> | REMOVE\_FILES <arrStr> | OP\_RES\_ARRAY <arrCode> |
|  |  |  |
| users | USERS | OP\_SUCCESSFUL <arrStr> |
|  |  | OP\_ERROR |
| trusted <trustedUserIDs> | TRUST\_USERS <arrStr> | OP\_RES\_ARRAY <arrCode> |
|  |  | OP\_ERROR |
| untrusted <untrustedUserIDs> | UNTRUST\_USERS <arrStr> | OP\_RES\_ARRAY <arrCode> |
|  |  | OP\_ERROR |
| download <userID> <file> | DOWNLOAD\_FILE <arrStr> | OP\_SUCCESSFUL <byteArray> |
|  | [0]=owner, [1]=fileName | ERR\_YOURSELF or ERR\_NOT\_FOUND or  ERR\_NOT\_FRIENDS or ERR\_NOT\_REGISTERED |
| msg <userID> <msg> | SEND\_MSG <arrStr> | OP\_SUCCESSFUL |
|  | [0]=receiver, [1]=textMsg | OP\_ERROR or ERR\_NOT\_FRIENDS  or ERR\_NOT\_REGISTERED |
| collect | COLLECT\_MSG <str> | OP\_SUCCESSFUL <arrListStr> |
|  |  | OP\_ERROR |
| exit | END\_CONNECTION | OP\_SUCCESSFUL |
|  |  |  |

A seguinte tabela apresenta os comandos e respectivas sintaxes, que os clientes podem requerer e a que o servidor pode responder.

Nas mensagens de pedido, a segunda linha corresponde à informação colocada no índex do array, enquanto que nas mensagens de resposta corresponde à mensagem de erro(a primeira linha corresponde à mensagem de resposta quando não há erro).

# Segurança

Os ficheiros tanto do servidor como do cliente estão “expostos”, ou seja, basta ir à directoria definida no programa, e consultar os ficheiros e mensagens de cada utilizador.

# Configuração da sandbox