README.md

SeiTchiz - An Instagram clone

Grupo 46

- Ana Albuquerque 53512
- Gonçalo Antunes 52831
- Tiago Cabrita 52741

Como compilar o projeto?

Abra o terminal dentro da pasta 'SegC-grupo46-proj1-2/SeiTchiz/' para executar os próximos comandos.

Servidor:

Navege para a pasta server: `cd server/libs/`

Coloque este comando no terminal: `java -jar Server.jar [Port] <keystore> <keystore-password>`

Cliente:

Navege para a pasta client: `cd client/libs/`

Coloque este comando no terminal: `java -jar Client.jar <IP>[:Port] <username> <truststore> <keystore> <keystore-password> <localUserID>`

Nota: Os campos contidos dentro de [ ] podem não ser colocados.

Keystores:

A keystore presente no servidor (server/keystore.server) utiliza a palavra-passe: ourserverpass. Qualquer uma das keystores do cliente (client/keystore.client) utilizam a mesma palavra-passe: ourclientpass.

Exemplo de utilização:

Servidor:

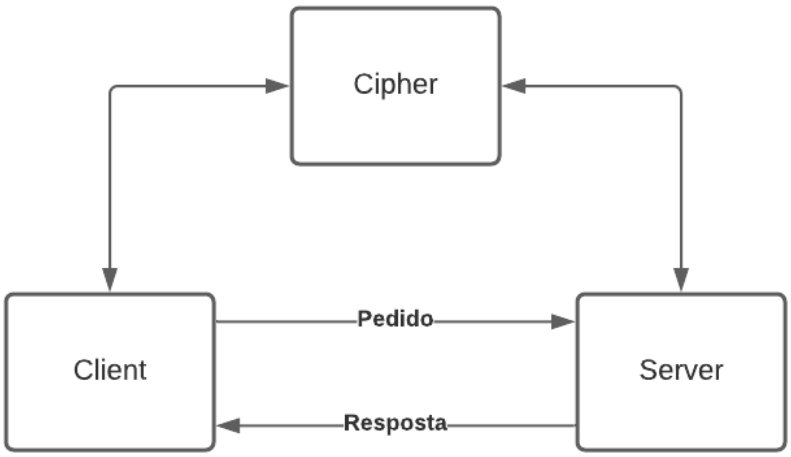
```
cd Fase1-Parte2/server/libs
java -jar Server.jar 45678 ../keystore.server ourserverpass
```

Cliente:

```
java -jar Client.jar 127.0.0.1:45678 ../truststore.client ../keystore.ana ourclientpass ana
```

Arquitetura do Software

O sistema SiTchiz pode-se dividir em dois, cliente e servidor. Contudo, de modo a facilitar o trabalho e a não repetir código foi ainda criado um módulo que trata das operações de cifra, decrifra, etc.



Cliente:

O cliente está organizado de forma a que haja uma classe que trate de todos os pedidos da conexão ao servidor. Os pedidos do cliente quando chegam são reencaminhados para um Handler (de pedidos - Request Handler) que tem como função reencaminhá-los para o Handler específico da operação a ser executada.

```
client
   | keystore.<nome do client> - guarda a chave privada do cliente no fomato JCEKS
   | truststore.client - trust store do cliente
   └──UserFiles
          | <nome da foto>.<extensão da foto> - foto que o cliente quer postar
```

Servidor:

O servidor está organizado de forma semelhante ao cliente. O processamento dos pedidos faz-se em 3 passos:

1. Receção do pedido;
2. Encaminhamento do mesmo para o Request Handler;
3. Request Handler envia para o Handler respetivo.

- Dentro da pasta `Data/` estão as pastas com as diversas informações relativas ao group; post; follow; publicKey e users;

Fizemos um esquema para melhor compreender a vista do projeto:

```
server
  └── Data
        └── follows
                └── <nome do user>
                        | follower.txt (*)
                        | following.txt (*)
                └── <nome do user example N> outra pasta
                        | ...
                        | N
        └── groups
                └── <nome do grupo criado> - Cada grupo tem a sua pasta
                                | groupinfo.txt - guarda os users do grupo e o id chave do grupo (*)
                                | groupkeys.txt - guarda a chave de cada user (*)
                        └── collect
                                | <nome do user>collect.txt - (1)
                        └── history
                                | <nome do user>history.txt - (2)
                └──  <group example N> outra pasta
                                | ...
                                | N
        └── posts
                └── <nome do user> - Cada user que postou tem a sua pasta
                                | <nome da foto postada>.<extensão da foto>
                                | <nome da foto postada>.txt - (3)
```

```
                            | <nome da foto postada>digest.txt - (4)
                        | nextID.txt - guarda o id da foto mais recente
                └─── <nome do user example N> outra pasta
                            | ...
                            | N
        └─── PubKeys
                | certSeiTchizServer.cer - certificado do servidor
                | <nome do user>.cer - certificado para um user do sistema
        └─── users
                | users.txt - guarda o par <UserId,nomeCerticado> cada vez que é autenticado um novo utilizador (*)
        | keystore.server - guarda a chave privada do servidor no fomato JCEKS
```

```
Nota: (*) significa que o ficheiro está cifrado
      (1) 'caixa' para onde vão as mensagens não lidas de cada utilizador antes de ser feito collect (*)
      (2) guarda as mensagens de cada utilizador depois de ter sido feito collect (*)
      (3) guarda a informação da foto, como os likes, o id e a extensão
      (4) guarda a síntese da foto para poder verificar a sua integridade (*)
```

Ao ser autenticado um novo utilizador é cifrado um novo par <UserId, nomeCerticado>, este é cifrado e guardado no ficheiro users.txt em `Data/users/`

Ao fazer follow o programa cria um diretorio com o nome do utilizador no directorio `server/Data/follows/` e dentro dele cria 2 ficheiros .txt um follows e um following, no following deve estar o nome do utilizador seguido;

Ao fazer unfollow o programa atualiza os ficheiros following do utilizador e o follower do user a deixar de ser seguido;

Ao fazer post, as fotos serão colocadas num diretório `server/Data/posts/<nome do utilizador que fez post>/`;

Ao fazer wall o programa retorna N posts dos seguidores do Utilizador e em caso de nao haver posts o programa informa o utilizador;

Ao fazer like o programa verifica a sintese da foto com o id dado, que foi guardada num ficheiro .txt, o programa atualiza o file .txt da foto a informação relativa aos likes;

Ao fazer newgroup o programa cria um ficheiro groupInfo.txt onde guarda os users do grupo e o id chave do grupo, e a chave cifrada de cada user fica guardada num ficheiro groupKeys.txt;

Ao fazer msg, as mensagens serão colocadas num diretório `server/Data/group/(nome do grupo)/collect/<nome de utilizador no grupo>collect.txt`;

Ao fazer collect, as mensagens serão colocadas num diretório `server/Data/group/(nome do grupo)/collect/<nome de utilizador no grupo>history.txt` e removidas do collect respectivo.

Cipher:

O cliente faz a encriptação e desencriptação das chaves e das mensagens; e o servidor do conteúdo dos ficheiros. Para isso foi criado o Cipher que é partilhado tanto pelo cliente como pelo servidor através de um jar. Com esta decisão aumentámos a escalabilidade do código uma vez que não irá haver código repetido no cliente e no servidor, pois ambos usam uma biblioteca que trata das operações de cifra e decifra.

Limitações:

- Na parte 1 fase 2, não conseguimos colocar o cliente e o servidor a correr numa sandbox.

Cliente:

- Por omissão a conexão é no porto 45678 (porto onde escuta o servidor);
- As respectivas keystores de cada cliente encontram-se na pasta `Fase1-Parte2/SeiTchiz/client/`;
- Para o cliente se autenticar com sucesso, é necessário que tenha na sua truststore o certificado do servidor, assim como o seu par de chaves assimétricas RSA na sua keystore;
- Para postar uma foto é necessário colocá-la na pasta UserFiles que está na pasta `client/`.
- Tem de verificar se está a correr pelo eclipse. Se não estiver, terá de a colocar na pasta `UserFiles` que está na pasta `libs` (onde está o jar);

- Para adicionar um utilizador a um grupo, é necessário que a chave pública do mesmo (no formato .cer) esteja presente no diretório `server/Data/publicKeys/`.

O cliente apenas reconhece os seguintes comandos/atalhos:

Comandos	Atalhos
follow [userID]	f [userID]
unfollow [userID]	u [userID]
viewfollowers	v
post [photo] *	p [photo]
wall [nPhotos]	w [nPhotos]
like [photoID]	l [photoID]
newgroup [groupID]	n [groupID]
addu/a [userID] [groupID]	a [userID] [groupID]
removeu [userID] [groupID]	r [userID] [groupID]
ginfo [groupID]	g
msg [groupID] [msg]	m [groupID] [msg]
collect [groupID]	c [groupID]
history [groupID]	h [groupID]
menu	--
quit	q

Servidor:

- Apesar de todos os ficheiros com informação sensível estarem devidamente cifrados, os nomes dos diretórios presentes em `Data/` revelam os nomes dos grupos.
- Apesar de todos os ficheiros com informação sensível estarem devidamente cifrados, os nomes dos ficheiros de certificados revelam os nomes dos users.
- Não é possível eliminar posts; grupos e mensagens.