Ciências ULisboa | Informática

di.ciencias.ulisboa.pt

/difcul

Segurança e Confiabilidade

2022/2023

Trabalho 1 – 1ª fase

1 Objetivos

A parte prática da disciplina de Segurança e Confiabilidade pretende familiarizar os alunos com alguns dos problemas envolvidos na programação de aplicações distribuídas seguras, nomeadamente a gestão de chaves criptográficas, a geração de sínteses seguras, cifras e assinaturas digitais, e a utilização de canais seguros à base do protocolo TLS. O primeiro trabalho a desenvolver na disciplina será realizado utilizando a linguagem de programação Java e a API de segurança do Java, e é composto por duas fases.

A primeira fase do projeto tem como objetivo fundamental a construção de uma aplicação distribuída, focando-se essencialmente nas funcionalidades da aplicação. O trabalho consiste na concretização do sistema ***Tintolmarket,*** que é um sistema do tipo cliente-servidor que oferece um serviço semelhante ao do *Vivino,* mas permitindo a compra e venda de vinhos por parte dos utilizadores do sistema. Existe uma **aplicação servidor** que mantém informação sobre os utilizadores registados e sobre vinhos listados, o seu valor, classificação e quantidade disponibilizada por cada utilizador. Os utilizadores podem adicionar vinhos, indicar quantidades que dispõem, classificar cada vinho, e ainda enviar mensagens privadas a outros utilizadores. Os utilizadores registados devem usar uma **aplicação cliente** para interagirem com o servidor e usar essas funcionalidades.

Nesta primeira fase do trabalho as questões de segurança da informação não são ainda consideradas, sendo apenas necessário concretizar as funcionalidades oferecidas pelo serviço. Na segunda fase do projeto o sistema será expandido para permitir compra de vinhos e serão adicionados requisitos de segurança relativos às comunicações e às aplicações. O enunciado da segunda fase será oportunamente divulgado.

2 Arquitetura do Sistema

O trabalho consiste no desenvolvimento de dois programas:

- O servidor *TintolmarketServer*
- A aplicação cliente *Tintolmarket* que acede ao servidor via *sockets* TCP

A aplicação é distribuída, uma vez que o servidor executa numa máquina e pode existir um número ilimitado de clientes a ser executados em máquinas diferentes na Internet.

3 Funcionalidades

A execução das aplicações deve ser feita da seguinte forma:

1. Servidor:

   TintolmarketServer <port>

   - <port> identifica o porto (TCP) para aceitar ligações de clientes. Por omissão o servidor deve usar o porto 12345.

2. Cliente:

Tintolmarket <serverAddress> <userID> [password]

Em que:

- <serverAddress> identifica o servidor. O formato de serverAddress é o seguinte: <IP/*hostname*>[:Port]. O endereço IP ou o *hostname* do servidor são obrigatórios e o porto é opcional. Por omissão, o cliente deve ligar-se ao porto 12345 do servidor.
- <clientID> identifica o utilizador local.
- [password] – password utilizada para autenticar o utilizador local. Caso a password não seja dada na linha de comando, deve ser pedida ao utilizador pela aplicação.

A aplicação cliente deve começar por enviar o *clientID* e a *password* ao servidor, para tentar autenticar o utilizador. Caso o utilizador não esteja registado no servidor, este fará o registo do novo utilizador, adicionando o *clientID* e a respetiva *password* ao ficheiro de utilizadores do servidor. Devem também ser inicializadas as estruturas de dados que mantêm informações relativas a cada utilizador. Em particular, dado que na 2ª fase do projeto será possível fazer compras, cada utilizador deverá terá um saldo mantido pelo servidor cujo valor inicial será 200.

Se o cliente não se conseguir autenticar no servidor (i.e., se o par *clientID/password* enviados não corresponderem aos que estão guardados no ficheiro de utilizadores do servidor), a aplicação deverá terminar, assinalando o erro correspondente. Caso contrário, a aplicação deverá apresentar um menu disponibilizando os seguintes comandos. Qualquer comando pode ser invocado por extenso ou pela letra que está a negrito (por exemplo, ***add*** ou ***a***):

- **a**dd <wine> <image> - adiciona um novo vinho identificado por *wine*, associado à imagem *image*. Caso já exista um vinho com o mesmo nome deve ser devolvido um erro. Inicialmente o vinho não terá qualquer classificação e o número de unidades disponíveis será zero. <wine> <image> <value> <quantity> <stars>

add ou replace

- **s**ell <wine> <value> <quantity> - coloca à venda o número indicado por *quantity* de unidades do vinho *wine* pelo valor *value*. Caso o *wine* não exista, deve ser devolvido um erro.

No replace passar arrayList e fazer ciclo while e dar replace em cada posição

- **v**iew <wine> - obtém as informações associadas ao vinho identificado por *wine*, nomeadamente a imagem associada, a classificação média e, caso existam unidades do vinho disponíveis para venda, a indicação do utilizador que as disponibiliza, o preço e a quantidade disponível. Caso o vinho *wine* não exista, deve ser devolvido um erro.

tres replaces em ficheiros diferentes

- **b**uy <wine> <seller> <quantity> - compra *quantity* unidades do vinho *wine* ao utilizador *seller*. O número de unidades deve ser removido da quantidade disponível e deve ser transferido o valor correspondente à compra da conta do comprador para o vendedor. Caso o vinho não exista, ou não existam unidades suficientes, ou o comprador não tenha saldo suficiente, deverá ser devolvido e assinalado o erro correspondente.
- **w**allet - obtém o saldo atual da carteira.

replace

- **c**lassify <wine> <stars> - atribui ao vinho *wine* uma classificação de 1 a 5, indicada por *stars*. Caso o vinho *wine* não exista, deve ser devolvido um erro.
- **t**alk <user> <message> - permite enviar uma mensagem privada ao utilizador *user* (por exemplo, uma pergunta relativa a um vinho à venda). Caso o utilizador não exista, deve ser devolvido um erro.

- read - permite ler as novas mensagens recebidas. Deve ser apresentada a identificação do remetente e a respetiva mensagem. As mensagens são removidas da caixa de mensagens do servidor depois de serem lidas.

O servidor mantém um ficheiro com os utilizadores do sistema e respetivas passwords. Este ficheiro deve ser um **ficheiro de texto**. Cada linha tem um *userID* e uma *password,* separados pelo caracter dois pontos (<userID>:<password>). As informações relativas aos vinhos e as caixas de mensagens de cada utilizador também devem ser mantidas em ficheiros (para além de poderem ser mantidas em memória), para evitar que se perca informação no caso de uma falha do servidor.

4 Entrega

Dia **24 de março**, até às 23:59 horas. O código desta primeira fase do trabalho deve ser entregue de acordo com as seguintes regras:

- O trabalho é realizado em grupos de 3 alunos, formados atempadamente.
- A formação de grupos é feita na página da disciplina no Moodle, até ao dia 17 de março.
- Para entregar o trabalho, é necessário criar um **ficheiro zip** com o nome **SegC-grupoXX-proj1-fase1.zip**, onde **XX** é o número do grupo, contendo:
  - ✓ o código do trabalho;
  - ✓ os ficheiros jar (cliente e servidor) para execução do projeto;
  - ✓ um readme (txt) indicando **como compilar** e **como executar** o projeto, indicando também as limitações do trabalho.
- O ficheiro zip é submetido através da atividade disponibilizada para esse efeito na página da disciplina no Moodle. Apenas um dos elementos do grupo deve submeter. Se existirem várias submissões do mesmo grupo, será considerada a mais recente.
- Não serão aceites trabalhos enviados por email. Se não se verificar algum destes requisitos o trabalho é considerado não entregue.