Ciências ULisboa | Informática

di.ciencias.ulisboa.pt

/difcul

Segurança e Confiabilidade

2021/2022

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 ***Trokos,*** que é um sistema do tipo cliente-servidor que oferece um serviço semelhante ao do MB WAY. Existe uma **aplicação servidor** que tem acesso às contas bancárias dos utilizadores registados, e que oferece diversas funcionalidades relacionadas com a movimentação dessas contas (transferências, pagamentos, pedidos de pagamento, etc.). Os utilizadores registados devem usar uma **aplicação cliente** para interagirem com o servidor e usar essas funcionalidades.

Para facilitar o desenvolvimento e teste do sistema *Trokos*, considera-se que é o próprio servidor que mantém informação sobre as contas bancárias dos clientes e o respetivo saldo, não sendo assim necessário implementar interfaces para acesso aos servidores dos bancos.

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 serão considerados requisitos de segurança, para que as interações e o sistema no seu todo sejam seguros. O enunciado da segunda fase será oportunamente divulgado.

2 Arquitetura do Sistema

O trabalho consiste no desenvolvimento de dois programas:

- O servidor *TrokosServer*
- A aplicação cliente *Trokos* 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:

   TrokosServer <port>

   - <port> identifica o porto (TCP) para aceitar ligações de clientes. Por omissão o servidor deve usar o porto 45678.

2. Cliente:

   Trokos <serverAddress> <userID> [password]

   Em que:

   - <serverAddress> identifica o servidor. O formato de serverAddress é o seguinte: <IP/*hostname*>[:Port]. O endereço IP ou *hostname* do servidor são obrigatórios e o porto é opcional. Por omissão, o cliente deve ligar-se ao porto 45678 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. Para além disso, o servidor deve ainda inicializar a conta bancária associada ao utilizador com um saldo inicial pré-determinado (por exemplo, 100). 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, ***balance*** ou ***b***):

- **b**alance – obtém valor atual do saldo da sua conta.
- **m**akepayment <userID> <amount> – transfere o valor *amount* da conta de *clientID* para a conta de *userID*. Se o cliente não tiver saldo suficiente na sua conta ou se o utilizador *userID* não existir, deve ser assinalado um erro correspondente a cada uma das situações.
- **r**equestpayment <userID> <amount> – envia um pedido de pagamento ao utilizador *userID*, de valor *amount*. O pedido deve ser colocado numa lista de pagamentos pendentes associada ao utilizador *userID*. Se o utilizador não existir assinala um erro.
- **v**iewrequests – obtém do servidor a sua lista de pedidos de pagamentos pendentes. Cada pedido é composto por um identificador único atribuído pelo sistema, pelo valor, e pela identificação do utilizador que fez o pedido.
- **p**ayrequest <reqID> – autoriza o pagamento do pedido com identificador *reqID*, removendo o pedido da lista de pagamentos pendentes. Se o cliente não tiver saldo suficiente na conta, deve ser retornado um erro. Se o identificador não existir ou se for referente a um pagamento pedido a outro cliente, também deve retornar um erro.
- **o**btainQRcode <amount> – cria um pedido de pagamento no servidor e coloca-o numa lista de pagamentos identificados por *QR code*. Cada pedido tem um *QR code* único no sistema, e está associado ao *clientID* que criou o pedido (a quem o pagamento será feito), e ao valor *amount* a ser pago. O servidor deverá devolver uma imagem com o *QR code*.
- **c**onfirmQRcode <QRcode> – confirma e autoriza o pagamento identificado por *QR code*, removendo o pedido da lista mantida pelo servidor. Se o cliente não tiver saldo suficiente na conta, deve ser retornado um erro (mas o pedido continua a ser removido da lista). Se o pedido identificado por *QR code* não existir também deve retornar um erro.
- **n**ewgroup <groupID> – cria um grupo para pagamentos partilhados, cujo dono (*owner*) será o cliente que o criou. Se o grupo já existir assinala um erro.

- **a**ddu <userID> <groupID> – adiciona o utilizador *userID* como membro do grupo indicado. Se *userID* já pertencer ao grupo ou se o grupo não existir deve ser assinalado um erro. Apenas os donos dos grupos podem adicionar utilizadores aos seus grupos, pelo que deverá ser assinalado um erro caso o cliente não seja dono do grupo.
- **g**roups – mostra uma lista dos grupos de que o cliente é dono, e uma lista dos grupos a que pertence. Caso não seja dono de nenhum grupo ou não seja membro de nenhum grupo, esses factos deverão ser assinalados.
- **d**ividepayment <groupID> <amount> – cria um pedido de pagamento de grupo, cujo valor total *amount* deve ser dividido pelos membros do grupo *groupID*. O pedido deve dar origem a pedidos individuais a serem colocados na lista de pedidos pendentes de cada membro do grupo. Quando todos os pedidos individuais forem pagos, o pedido de grupo pode ser movido para um histórico de pedidos de grupo. Caso não seja dono do grupo ou o grupo não exista, deve ser assinalado um erro.
- **s**tatuspayments <groupID> – mostra o estado de cada pedido de pagamento de grupo, ou seja, que membros de grupo ainda não pagaram esse pedido. Caso não seja dono do grupo ou o grupo não exista, deve ser assinalado um erro.
- **h**istory <groupID> – mostra o histórico dos pagamentos do grupo *groupID* já concluídos. Caso não seja dono do grupo ou o grupo não exista, deve ser assinalado um erro.

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>). Os pedidos de pagamento pendentes (individuais e de grupo) e o histórico de pagamentos de grupo também têm de ser guardados em disco no servidor (para que, caso necessário, seja mais fácil lidar com falhas do servidor sem perder informação).

Na implementação do comando obtainQRcode, o servidor tem de gerar um código (pode ser uma *string*) que identifique o pedido de pagamento feito pelo cliente (que na realidade é tipicamente uma loja). Este código pode ser representado por um *QR code* (https://en.wikipedia.org/wiki/QR_code), que é um elemento gráfico (uma imagem). No contexto deste trabalho, admite-se que o servidor use um conjunto de códigos e respetivos *QR codes* previamente definidos e armazenados no sistema de ficheiros. Contudo, será mais interessante e os trabalhos serão mais valorizados se códigos únicos e os respetivos *QR codes* forem gerados dinamicamente (por exemplo, usando bibliotecas disponíveis na Web, como a biblioteca ZXing).

4 Entrega

Dia **25 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 11 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.