
Escola de Ciências e Tecnologia

Licenciatura em Engenharia Informática

Sistemas Operativos II

# *Segundo trabalho pratico*

*Autores:*

João Martins
27396
Miguel Casco
28966

June 28, 2015

# Contents

# 1 Introdução

Este trabalho consiste em fazer uma revisão do sistema de vendas do aeroporto, feito no primeiro trabalho. Iremos ter uma WebApp a que o cliente terá acesso, ligada à base de dados na plataforma alunos.di.uevora.pt. O cliente irá ter acesso a pesquisa de voos, a compra de bilhetes para voos, sendo emitido um recibo, e a consulta da lista de passageiros de cada voo.

# 2 Base de dados e replicação

Tal como no primeiro trabalho, a nossa base de dados consiste em 2 tabelas, voo e venda. Fizemos uma modificação na venda de modo a facilitar a emissão de recibos, que consiste em a chave primária serem todos os campos. Deste modo faz com que a venda tenha o id_venda que será o numero do recibo, o id que será o voo referente a venda e o nome do passageiro.
Em matéria de replicação, optamos por replicação passiva, sendo que fizemos mais duas tabelas iguais a venda e duas iguais a voo (fazendo um total de 6 tabelas). Quando existe uma inserção na tabela voo ou venda, essa inserçãoo é replicada para as outras duas copias, assegurando que se houver um problema nos dados eles estão guardados em mais dois locais.

# 3 Aplicação Servidor

O nosso servidor tem uma interface remota (Voos), que fornece metodos remotos para serem utilizados pela WebApp. Vai estabelecer uma conexão à base de dados da plataforma alunos.di.uevora.pt, autenticando o utilizador da base de dados, e dependente do pedido do cliente através da WebApp, vai executar uma query (preparada para PostGres SQL) sobre a base de dados. Dependendo ainda da query os dados podem ser trabalhados, como por exemplo no caso da venda, em que é necessario fazer um update no numero de lugares disponiveis e inserir novos dados, ou apenas devolver os dados para o cliente, como no caso de mostar os voos para um determinado destino. É ainda o servidor que trata da replicação, fazendo no caso da compra que é a opção que altera a a base de dados a pedido da WebApp, os inserts correspondentes nas tabelas.

# 4 WebApp

A nossa aplicação WebbApp é onde o cliente coloca a acção que pretende que o servidor faça. Após a acção ser escolhida, a mesma será procurada no serviço de nomes, que tem de ter sido lançado antes (comando: rmiregistry -J-classpath -Jclasses:. 9000). Depois de invocar um metodo do objeto remoto o nosso servidor executará a query apropriada e devolverá a informação devida ao cliente. Em termos de controlo de concorrencia, consideramos que apenas a compra deve ser concorrente, porque é a que altera dados. Tanto a pesquisa como a consulta, não tem problemas em que várias pessoas utilizem ao memso tempo dado que são apenas selects e não inserts. No caso da compra fizemos com que fosse synchronized, sendo que quando alguem esta a tentar comprar o metodo synchronize vai bloquear até a compra estar efectuada e só depois outro cliente poderá comprar.

# 5 Emissão dos recibos

Após ser efectuada uma compra será emitido um recibo. Vai ser criado um ficheiro recibo_N, enviado ao cliente, e um ficheiro digest_N que vai ser o recibo encriptado. Para a enccriptação do recibo foi utilizado SHA-1 com MessageDigest do Java. O algoritmo foi escolhido após alguma pesquisa, pois ele utiliza 160 bits para a encriptação.

# 6 Makefile

O trabalho foi feito em netBeans e para correr em netBeans. O cuidado que se tem de ter é lançar o serviço de nomes antes,
que é feito executando o comando,

rmiregistry -J-classpath -Jclasses:. 9000
na pasta correspondente. O servidor é lançado atraves do botão direito-run.
A webbApp é so fazer play.