Universidade do Minho

Licenciatura em Engenharia Informática

Relatório de Sistemas Distribuídos

Grupo 30

Unidade Curricular: Sistemas Distribuídos [J305N4]

Coordenador: Professor José Orlando Roque Nascimento Pereira

Nome: Carlos Filipe Almeida Dias

Número: 93185

Contacto: a93185@alunos.uminho.pt

Curso: Licenciatura em Engenharia Informática, Universidade do Minho

Nome: José Pedro Martins Magalhães

Número: 93273

Contacto: a93273@alunos.uminho.pt

Curso: Licenciatura em Engenharia Informática, Universidade do Minho

Nome: Francisco Reis Izquierdo

Número: a93241

Contacto: a93241@alunos.uminho.pt

Curso: : Licenciatura em Engenharia Informática, Universidade do Minho

Nome: Duarte Augusto Rodrigues Lucas

Número: a89526

Contacto: a89526@alunos.uminho.pt

Curso: : Licenciatura em Engenharia Informática

Ano Letivo 2021/2022

Conteúdo

Índice de Tabelas

1 Introdução

Este projeto proposto pelos docentes da unidade curricular de Sistemas Distribuídos, teve como objetivo o desenvolvimento de uma plataforma de reserva de voos, sob o modelo cliente-servidor em Java que pretende fornecer recursos e serviços para um ou mais clientes, utilizando apenas uma socket e várias threads.

Ao longo deste relatório especificamos os métodos seguidos e as diversas decisões tomadas com o fim de se ter um serviço que comunique com o cliente permitindo-o reservar viagens diretas ou viagens em escala (constituição de vários voos), sendo que o servidor deverá informar ao cliente que ao fazer uma reserva, esta é feita com sucesso.

2 Arquitetura

2.1 Servidor

A classe servidor é a classe responsável por estar constantemente a ouvir e aceitar pedidos de conexão TCP, na porta 12345, de clientes ou administradores que são depois direcionados para a classe "serverWorker" que lida com eles concorrentemente. Para além disto, esta classe é também responsável por manter controlo do número de threads "serverWorker" que estão a correr através de uma variável global "threads", isto permite ao servidor esperar que todas terminem assim que a socket é encerrada, não interrompendo nenhuma operação de um cliente ou administrador a meio nem aceitar novos pedidos de conexão, isto permite também garantir que a nossa base de dados apenas é guardada (serializada em ficheiro) depois de todas as threads terminarem evitando erros de concorrência e IO.

2.2 ServerWorker

A classe do *serverWorker* é a thread utilizada pelo servidor para processar os vários pedidos de um dado cliente ou administrador, ficando sempre conectada até que o seu utilizador encerrar a conexão. Esta classe é responsável por fazer o *parsing* do pedido do utilizador, disponibilizando o menu adequado que requer *input* do mesmo. Cada instância desta classe é controlada concorrentemente através de um mecanismo de controlo de concorrência.

2.3 Cliente

Cada cliente tem de efetuar uma autenticação, digitando o seu username e password, ou se for a situação de uma primeira vez, deverá registar-se dando um username e uma password, mas caso este username já se encontre na base de dados, o cliente será notificado que o registo não foi efetuado com sucesso. Após efetuar o login, terá diversas opões de escolha, tais como, cancelar uma reserva num voo, usando o código de reserva caso o dia não se encontre encerrado. Poderá também obter uma lista de voos existentes e fazer uma reserva, inserindo a

respetiva origem e destino da sua preferência, fazendo com que o servidor gere uma lista de destinos possíveis com a origem, como também uma lista de voos possíveis desde a origem até ao destino, para que assim o cliente faça a sua escolha.

2.4 Administrador

O administrador tem de efetuar uma autenticação, digitando o seu username e password, o sistema por sua vez, verificará se o username e password introduzidos correspondem, caso contrário o sistema notifica o administrador que os dados são incorretos. Após efetuar o login, terá diversas opões de escolha, tais como, inserir um voo, encerrar um dia, listar voos e encerrar o servidor.

2.5 BaseDados

A classe BaseDados é responsável por armazenar toda a informação gerada pelos utilizadores. Esta contém informações dos usernames e as correspondentes palavras-passe de todos os clientes, de todos os voos disponíveis, dos dias encerrados e de todas as reservas efetuadas. De modo a manter controlo de concorrência a nível dos acessos às diferentes estruturas de dados, foram implementados vários locks visando a manutenção da coerência dos dados permitindo acesso apenas a um cliente de cada vez a certos tipos de dados.

A base de dados permite guardar os dados gerados de forma persistente num ficheiro de objetos através da implementação da interface "Serializable".

Por fim, uma vez que não permitimos o registo de novos administradores, a instância da classe é inicializada de forma automática se não estiver guardada num ficheiro já com um administrador por default.

3 Concorrência

A concorrência revelou um papel importante no controlo de acesso à base dados por parte dos clientes e do administrador, sendo por esse mesmo motivo que a implementação de locks foi feita na classe BaseDados. Cada serverWorker foi responsável também por implementar concorrência.

4 Frames

A serialização dos dados tem como objetivo enviar dados por parte do cliente, administrador e serverWorker, em que estes podem ser pedidos, como também respostas, por parte das classes referidas.

Na desserialização trata-se de um processo inverso à serialização, ou seja, na obtenção do conteúdo dos dados recebidos esta é feita por parte do serverworker, cliente e administrador, como já mencionados. No serverworker, a desserialização tem uma função importante, uma vez que existem tags associados às perguntas efetuadas pelo administrador e os pelos diversos clientes.

Tag	Descrição
0	Autenticar cliente
1	Registar cliente
2	Autenticar administrador
3	Fazer reserva
4	Encerrar dia
5	Adicionar voo
6	Listar voos
7	Cancelar voo
8	Encerrar servidor

Tabela 1: Tags

5 Funcionalidade Adicionais

Enquanto grupo, decidimos também implementar umas das duas funcionalidades adicionais propostas no enunciado, sendo esta a obtenção de uma lista com todos os percursos possíveis para viajar entre uma origem e um destino, limitados a duas escalas (três voos), minimizando a quantidade de dados transferidos.

6 Conclusões

A unidade curricular de Sistemas Distribuídos tem como objetivo apresentar-nos conceitos que nos levam a melhorar o conceito de sistemas distribuídos, como também saber dividir tarefas e responsabilidades. Na conceção deste projeto deu-nos uma visão mais abrangente de uma implementação de um sistema entre cliente-servidor em java, em que usamos os conceitos abordados nas aulas, tais como o controlo de concorrência no acesso de informação, a utilização de threads e serialização e desserialização de dados em sistemas distribuídos. De um modo geral este projeto permitiu-nos criar bases seguras e que poderão ser importantes na área de desenvolvimento de sistemas distribuídos.