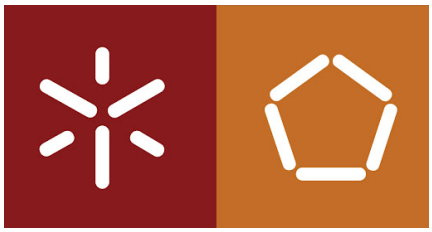

Universidade do Minho

Licenciatura em Engenharia Informática

Reserva de Voos - Sistemas Distríbuidos
Grupo 40

Bernardo Emanuel Magalhães Saraiva - A93189

José João Cardoso Gonçalves - A93204

Moisés Araújo Antunes - A82263 Rui Filipe Coelho Moreira - A93232

Ano Lectivo 2021/2022

1 Introdução e Objetivos do Trabalho

O presente relatório visa apresentar o projeto desenvolvido no instrumento de avaliação em grupo da UC de Sistemas Distribuídos, explorando os conceitos de programação concorrente e a relação cliente-servidor, recorrendo à utilização de *sockets* e *threads*. A temática proposta surge no desenvolvimento de uma plataforma de reserva de voos que permite a reserva e consulta de viagens. Para o efeito, o projeto consiste numa plataforma que permite aos utilizadores, depois de autenticados, a reserva de viagens, sendo estas constituídas por vários voos, informando os clientes quando a reserva é realizada com sucesso e permitindo o seu cancelamento atempado.

A aplicação desenvolvida permite que seja utilizada por vários utilizadores em simultâneo, tornando-se necessário recorrer a métodos de controlo de concorrência para garantir que os pedidos sejam atendidos atempadamente e correctamente, sendo todas as respostas devolvidas aos utilizadores correctos. Os utilizadores recorrem a uma autenticação para aceder à plataforma e usufruir de todas as funcionalidades do sistema. O sistema conta também com um administrador que possui funcionalidades distintas, nomeadamente inserção de informações de voos (origem, destino e capacidade) e o encerramento do sistema, deixando de permitir reservas para o dia em questão e cancelando as existentes.

Relativamente ao utilizador, após a autenticação na plataforma, este tem a liberdade de consultar a lista dos voos existentes por origem e destino, assim como reservar e cancelar uma das suas viagens. Para efetuar uma reserva, o utilizador deve indicar a origem, o destino e a data relativas à viagem, sendo que o sistema consulta as informações de voos correspondentes à data indicada e, caso haja disponibilidade de voo, gera um código de reserva que permite o utilizador consultar e cancelar a viagem, caso pretenda.

Deste modo, através do presente relatório explicar-se-á os vários componentes da aplicação desenvolvida, assim como todas as suas funcionalidades e a sua arquitectura.

2 Arquitetura do Projeto

Como forma de organização, o presente projeto foi dividido em 3 packages principais, pelo que se passa a descrever a função de cada um:

- Client: neste package encontram-se todas as classes relativas à conexão do cliente e à passagem das suas mensagens para a área do servidor;
- Model: o package do Model contém as estruturas básicas de dados, assim como os métodos auxiliares para cálculos e manipulação da informação que é feita pelo servidor.
- Server: o package Server contém o servidor e os métodos relativos ao *handling* de conexões e transmissão das mensagens.

3 Descrição das classes Implementadas

Uma vez que o projeto se concentra na comunicação cliente-servidor, começou-se por desenvolver ambas as classes, tendo assim uma base do projeto, tornando-se mais fácil trabalhar e desenvolver a partir deste ponto as restantes funcionalidades.

3.1 Servidor

A classe Servidor implementa a classe ServerSocket, que se mantém à escuta e, assim que obtém uma ligação, cria um novo Socket que aceita e recorre à classe ServerConnection para tratar o pedidos que recebe. Na sua execução, recorre também à classe Info, que trata a informação. É de ressaltar que esta classe permite que haja uma diversidade de pedidos em simultâneo a ser tratados, uma vez que recorre a uma nova *thread* para cada pedido que é aceite.

3.2 Cliente

O Cliente é responsável pela interface que vai permitir ao utilizador submeter as informações e enviá-las de modo a serem manipuladas e tratadas no servidor.

3.3 Account

O sistema requer a autenticação e registo de utilizadores através de um *username* e uma *password*, permitindo assim a interacção com o serviço. A classe Account contém todas as informações de um utilizador, como o seu ID e *password*, bem como as listas de reservas que o utilizador fez. Adicionalmente, contém um boolean que, no caso de estar a *true*, indica que a conta pertence a um administrador, caso este em que lhe são permitidas funcionalidades distintas das que os utilizadores possuem.

3.4 Info

A classe Info guarda as informações relativas aos voos, contas de utilizadores e dias cancelados, pelo que se passa a descrever em seguida de acordo com os nomes dados:

- flightsMap: Armazena o conjunto dos voos, sendo que tem como chave a origem do voo, e como value a lista dos voos que partem dessa origem. Assume-se que os mesmos voos se repetem diariamente, tal como descrito no enunciado;
- closedScheduleList: Lista em que se guarda os dias cancelados através de um *LocalDate*, sendo o encerramento de um dia determinado pelo Administrador;

- accountsMap: Este Map armazena as informações dos utilizadores, tendo como *key* o nome de utilizador e a correspondente Account como *value*;

É de referir que as informações mencionadas anteriormente implementam todas a persistência de dados, sendo que as mesmas são guardadas em formato binário na pasta *saves*.

Para além de poder editar as informações anteriormente descritas, os métodos desta classe permitem também criar e entrar em contas, atualizar a origem, o destino e a capacidade de cada voo, marcar, cancelar e verificar voos, fechar em certos dias e calcular uma lista de percursos possíveis, dada uma origem e um destino.

3.5 Flight

A classe Flight permite definir, guardar e atualizar os dados do voo, nomeadamente o destino, a capacidade máxima e os lugares ocupados para cada data. Também determina e permite verificar quantos lugares vagos restam no voo, de modo a que os utilizadores possam saber se ainda é possível reservar viagens para esse voo. Esta classe tem como variável de instância um Map que, para cada dia em que haja viagens, tem o número de viagens que estão marcadas. Conta também com a capacidade e o destino do voo.

3.6 Reservation

A classe Reservation possui os detalhes de cada reserva, nomeadamente o dia, o trajeto da viagem e um boolean que diz se a reserva foi cancelada.

3.7 ServerConnection

A classe ServerConnection é a representação de cada sessão, sendo responsável por interpretar o input proveniente do cliente recorrendo a uma TaggedConnection. Esta classe funciona como um Handler.

3.8 TaggedConnection

A classe TaggedConnection permite a troca de mensagens entre cada uma das partes recorrendo a Data[Input—Output]Stream's, sendo portanto um intermediário agnóstico ao conteúdo das mesmas. Para permitir a associação entre pedidos e respostas, a TaggedConnection utiliza uma Tag (explorada na classe Frame) que permite identificar a funcionalidade correspondente ao Frame em questão, seguido dos dados da mensagem.

3.9 Frame

Como forma de simplificar as mensagens que são enviadas/recebidas, estas foram representadas recorrendo a frames, que são constituídos por uma Tag, seguido dos dados que compõem a mensagem. Deste modo, passamos a demonstrar o significado de cada uma das Tag's.

Tag	**Significado**
LOGIN	Pedido/resposta de início de sessão
SIGNUP	Pedido/resposta de registo de um usuário
LOGOUT	Pedido/resposta de término de sessão
INSERT_FLIGHT	Pedido/resposta de inserção de um novo voo
CLOSE_DAY	Pedido/resposta de fecho do calendário de voos para um determinado dia
CLOSE_SERVICE	Pedido/resposta de fecho da conexão do servidor
BOOK_TRIP	Pedido/resposta de reserva de uma viagem
CANCEL_FLIGHT	Pedido/resposta de cancelamento de uma reserva
GET_FLIGHTS_LIST	Pedido/resposta de obter a lista de voos
GET_ALL_ROUTES	Pedido/resposta de obter uma lista com todos os percursos possíveis para viajar entre uma origem e um destino
STRESSED	Pedido/resposta que verifica se o servidor se encontra sobrecarregado
RESERVATIONS	Pedido/resposta para obtenção da lista de reservas do utilizador

3.10 Demultiplexer

A classe Demultiplexer surge como uma abstração para que seja possível receber várias mensagens ao mesmo tempo, bem como para enviar pedidos para o servidor. Através do método start desta classe o cliente fica constantemente à escuta numa thread, armazenando os dados recebidos de acordo com as suas Tags num Map <Tag,FrameValue>. A classe FrameValue possibilita o armazenamento de várias mensagens por Tag, implementando para isso uma Queue de arrays de bytes com o conteúdo das diversas mensagens ainda não tratadas. Através do método receive, é possível retirar deste Map a mensagem que pretendemos tratar no momento e esperar pela receção da mesma caso esta ainda não tenha sido recebida.

4 Demonstração de Resultados

Após testar o programa implementado e todas as suas funcionalidades desenvolvidas (obrigatórias e adicionais), verificou-se que os resultados estavam a ser dispostos como pretendidos, pelo que se demonstra em seguida alguns dos menus, operações e resultados:

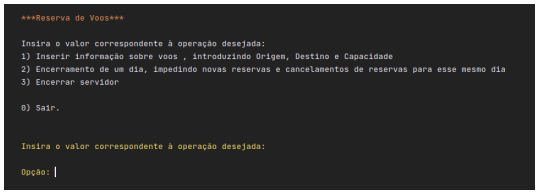

Figura 1: Menu Admin

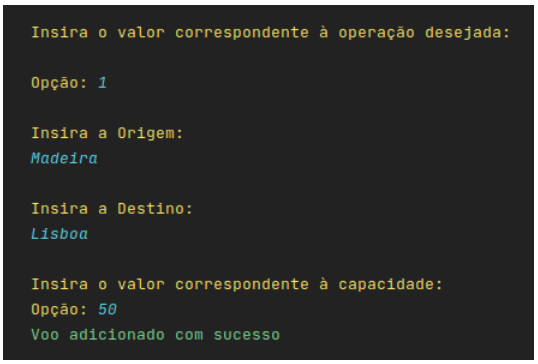

Figura 2: Funcionalidade do Admin - Inserir voo

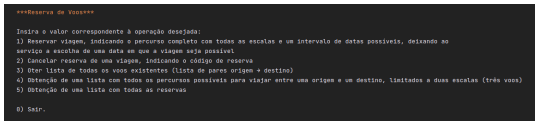

Figura 3: Menu Cliente

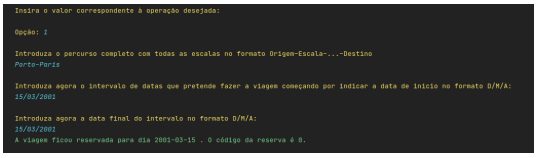

Figura 4: Funcionalidade do Cliente - Reservar viagem

5 Conclusão

Com a realização do presente projeto, conclui-se que foi possível obter uma maior percepção do modo como a programação de sistemas distribuídos com recurso ao par cliente-servidor, sockets e thread's funcionam. Para além disso, o projeto permitiu identificar e combater problemas do mundo real no contexto de Sistemas Distribuídos, nomeadamente os problemas relacionados às secções críticas e a utilização de sockets para a transmissão de informação.

De um modo geral, pensa-se que é apresentado um projeto sólido, que cumpre com as funcionalidades e os objetivos propostos, assim como os adicionais, destacando-se a capacidade de efetuar a comunicação cliente-servidor de maneira responsiva e eficaz, permitindo a gestão e reserva de voos.

Para trabalho futuro, seria interessante implementar a persistência com recurso a Bases de Dados, assim como o uso de uma arquitetura MVC, caso se expandisse o projeto, uma vez que não se verificam grandes vantagens dadas as dimensões do projeto.