Universidade do Porto

Faculdade de Engenharia

FEUP

Sistemas Distribuídos

3º ano do Mestrado Integrado em Engenharia Informática e Computação

UPER

Relatório Final

Bernardo Leite - up201404464@fe.up.pt
Francisca Cerquinho - up201505791@fe.up.pt
Luís Saraiva - up201404302@fe.up.pt
Verónica Fradique - up201506440@fe.up.pt

26 de Maio, 2018

Contents

1 Introdução **2**

2 Funcionalidades **3**

3 Arquitetura **4**
3.1 Servidor 5
3.2 Cliente 5

4 Implementação **6**
4.1 Concorrência 6
4.2 Mensagens 6

5 Assuntos Importantes **7**
5.1 Tolerância a falhas 7
5.2 Escalabilidade 7
5.3 Notificações 7
5.4 Segurança 8

6 Conclusão **9**

1 Introdução

UPER é um Sistema Distribuído que tem como principal função a partilha de boleias entre as pessoas envolvidas na comunidade da Universidade do Porto. Com este projeto pretendemos implementar uma plataforma capaz de gerir viagens entre dois pontos. Desta forma, um utilizador registado que vai realizar uma viagem poderá inseri-la na plataforma e outro utilizador poderá associar-se a esta viagem.

Para garantir que é um sistema exclusivo para membros da comunidade UP o login e o registo no sistema é restringido a emails que sejam do domínio da UP.

Neste relatório, irá ser descrito o projeto da seguinte forma: inicialmente falar-se-á sobre a **arquitetura** da aplicação, posteriormente serão clarificados os detalhes mais importantes da **implementação** do trabalho, seguido de uma descrição da abordagem adotada no âmbito da **segurança, tolerância a falhas, escalabilidade** e **consistência**.

Por fim, concluiremos este relatório com uma análise crítica do nosso trabalho assim como possíveis melhoramentos que o grupo considera exequíveis e pertinentes.

2 Funcionalidades

Ao iniciar a aplicação temos o menu de autenticação, onde o utilizador poderá fazer login ou registar-se.

- Criar uma viagem. O utilizador cria uma viagem definindo a hora e local de partida e destino, assim como o número de lugares que tem disponíveis.

- Apagar uma viagem. O utilizador poderá apagar qualquer uma das suas viagens.

- Gerir as suas viagens. O utilizador poderá juntar ou remover passageiros que tenham demonstrado a intenção de se juntar à sua viagem.

- Procurar viagens. O utilizador poderá procurar por viagens, inserindo data, hora e local de partida e chegada.

Listar viagens desejadas. Nesta opção, são listadas todas as viagens que não existem mas que o utilizador deseja realizar.

- Juntar-se a uma viagem. O utilizador pode pedir para se juntar a uma viagem, esta aprovação é sempre feita pelo seu criador.

- Sair de uma viagem. O utilizador pode pedir para sair de uma viagem à qual se juntou.

- As minhas viagens. O utilizador poderá listar todas as viagens que criou.

- As viagens às quais me juntei. O utilizador poderá listar todas as viagens a que se juntou.

- Esperar por uma viagem. O utilizador poderá declarar a intenção de se juntar a uma viagem que ainda não existe. Este recebe uma notificação caso alguém crie uma viagem que se assemelhe a este perfil.

Se o utilizador for um administrador, o menu terá mais duas opções:

- Visualizar todas as viagens existentes;

- Apagar qualquer uma das viagens da aplicação.

3 Arquitetura

Para uma comunicação fluente entre cliente e servidor, foi criado um conjunto de protocolos e conexões, de modo a garantir a disponibilidade do serviço e a respectiva tolerância a falhas.

Relativamente aos canais de comunicação utilizados, foram criados quatro canais: *"AuthenticationChannel", "OwnerChannel", "JoinTravelChannel"* e *"ListChannel"*.

AuthenticationChannel

Este canal é responsável pela troca de mensagens que concretiza o processo de autenticação. Desta forma, é através deste canal que se faz o registo e login de um dado utilizador. Note-se que na troca de mensagens deste canal utiliza-se a tecnologia de segurança SSL para estabelecer uma ligação encriptada entre um cliente e o servidor e certificados SSL para autenticar as identidades.

OwnerChannel

Este canal é responsável pelo envio e receção das mensagens relativas à criação e eliminação de uma determinada viagem.

JoinTravelChannel

Este canal é responsável pelo envio e receção das mensagens que permitem ao utilizador juntar-se ou sair de uma determinada viagem.

ListChannel

Este canal é responsável pelo envio e receção das mensagens que permitem ao utilizador listar as viagens disponíveis da aplicação mediante os filtros de pesquisa que forem aplicados no momento.

A arquitetura da aplicação desenvolvida é do tipo Cliente/Servidor. O projeto encontra-se dividido por diversos ficheiros, dos quais se destacam:

- Server.java: classe responsável por toda a atividade do servidor, que inicializa o mesmo com o fim de comunicar com os diversos clientes.

- Client.java: classe responsável por gerir todas as funcionalidades atribuídas ao cliente.

- MessageTreatment.java: responsável pelo tratamento das mensagens e envio da mensagem da resposta, que poderá ser de sucesso ou insucesso. No caso do cliente enviar uma mensagem com o intuito de listar, a mensagem de resposta será a respetiva lista.

- Messages.java: contém todas as mensagens que podem ser trocadas entre o cliente e servidor para realizar todo o tipo de ações;

- User.java: classe que gere um utilizador;

- Travel.java: classe que gere uma viagem.

3.1 Servidor

O Servidor tem como principal funcionalidade responder aos pedidos feitos pelos Clientes. Eis as suas principais funcionalidades :

- Estabelecer os canais de comunicação;
- Registar um utilizador na aplicação;
- Criar/Apagar uma viagem;
- Pesquisa/Listagem de viagens segundo determinados parâmetros;
- Juntar/Remover um utilizador a uma viagem;
- Enviar notificações aos utilizadores;
- Responder a todos os pedidos de um Cliente.

3.2 Cliente

O Cliente tem como principal funcionalidade criar pedidos para serem enviados ao Servidor. Eis as suas principais funcionalidades:

- Estabelecer os canais de comunicação;
- Apresentar menus interativos que permitem ao utilizador interagir com a aplicação;
- Validação da inserção dos dados pelo utilizador;
- Efetuar os pedidos ao Servidor.

A classe "Client" contém os sucessivos menus a que o cliente pode aceder para realizar todas as ações que o programa permite. Para além disso, também tem todas as validações para os campos que o utilizador tem de introduzir, assim como as respetivas mensagens de erro para quando uma informação não é introduzida com o formato esperado. Nestas situações, é pedido ao utilizador para reintroduzir os dados. Para cada ação é criada a mensagem que depois é enviada, por este, para o respetivo canal de comunicação para que se proceda à comunicação com o servidor.

4 Implementação

O projeto UPER pode ser dividido em duas grandes partes, Servidor e Cliente. Ambos estão implementados com a linguagem Java, dado que era um dos requisitos a cumprir e também se revelou uma mais valia, tendo em conta que o primeiro projeto já tinha sido implementado na mesma linguagem.

4.1 Concorrência

De forma a permitir que o programa execute várias tarefas ao mesmo tempo foi necessário lidar com problemas de concorrência. Desta forma, são utilizadas diferentes threads, que dependendo do tipo de ação, permitem que não haja perdas de informação devido ao excesso de informação a circular em ambos os sentidos - cliente/servidor e vice versa. No nosso caso é criada uma Thread para cada uma dos canais de comunicação, representados pelas classes AuthenticationChannel, OwnerChannel e JoinTravelChannel, ListChannel. Nos casos em que se aguarda confirmação da receção de uma mensagem, com objetivo de não sobrecarregar um determinado canal, dá-se um tempo determinado entre o envio de múltiplas mensagens seguidas.

4.2 Mensagens

As mensagens enviadas aos utilizadores seguem a seguinte estrutura:
<Ação Realizada><Email do destinatário><body>

Seguem os seguintes exemplos:

- Register <email><password>
- Login <email><password>
- Success <email><registration>
- Failed <email><registration>
- Create <email><date><startPoint><endPoint><numberOfSeats>
- Delete <email>travel <travelID>
- Join <email>**travel number** <travelID>
- Leave <email>**travel number** <travelID>
- AddPassenger <email>**travel number** <travelID>**passenger** <emailPassenger>
- RemovePassenger <email>**travel number** <travelID>**passenger** <emailPassenger>
- SearchComplete <email ><day><startPoint><endPoint>
- SearchPartial <email ><day><startPoint><endPoint>
- ListJoinTravels <email>
- ListRequestTravels <email>

5 Assuntos Importantes

5.1 Tolerância a falhas

Qualquer campo que seja introduzido incorretamente pelo utilizador, o sistema não o aceita e é apresentada uma mensagem de erro, para que o utilizador volte a introduzir.

No caso de o cliente ou servidor se desconectarem da internet é enviada uma mensagem de erro e o programa termina.

5.2 Escalabilidade

Para tornar a aplicação facilmente escalável, foi criada uma estrutura de dados que guarda toda a informação dos utilizadores. Desta forma, os dados são facilmente acessíveis. A aplicação também é escalável na medida em que é de fácil manutenção e modificação de funcionalidades. Por fim, a arquitetura implementada é capaz de hospedar quer um número restrito de utilizadores assim como um elevado número.

5.3 Notificações

Implementamos na nossa aplicação um vasto conjunto de notificações, descritas abaixo, de modo a aumentar a usabilidade da mesma.

- "NotificationAddChannel"

Um utilizador recebe uma notificação quando é aceite como passageiro numa viagem.

- "NotificationCreateTravelChannel"

Um utilizador recebe uma notificação quando uma viagem é criada e é semelhante a uma das suas viagens desejadas.

- "NotificationDeleteTravelChannel"

Um utilizador recebe uma notificação quando uma viagem à qual ele se juntou é apagada.

- "NotificationExitTravelChannel"

Um utilizador recebe uma notificação quando o criador da viagem o removeu como passageiro.

- "NotificationJoinChannel"

O utilizador recebe uma notificação quando o criador da viagem o adicionou como passageiro.

- "NotificationLeaveChannel"

O utilizador recebe uma notificação quando algum passageiro saiu da sua viagem.

Quando um utilizador inicia uma das ações indicadas acima, o servidor despoleta uma thread que tem como finalidade enviar uma mensagem para o(s) utilizador(es) a que a mensagem se destina. Esta mensagem é enviada até que o(s) respetivo(s) utilizador(es)

envie(m) uma mensagem de confirmação. Após o login de um utilizador, são criados os canais de comunicação que irão estar à escuta das mensagens de notificação enviadas pelo servidor.

Os utilizadores são notificados com a seguinte estrutura de mensagem:
<Tipo de notificação><email do destinatário><body>

Seguem as seguintes notificações:

- NotificationJoinTravel <email>**travel number** <travelID>**passenger** <email>

- NotificationExitTravel <email>**travel number** <travelID>**by** <emailCreator>

- NotificationDeleteTravel <email>**travel number** <travelID>**by** <emailCreator>

- NotificationAddPassenger <email>**travel number** <travelID>**by** <emailCreator>

- NotificationLeaveTravel <emailCreator>**travel number** <travelID>**passenger** <email>

- NotificationCreateTravel <email>**travel number** <travelID>

A confirmação da receção da notificação tem a seguinte estrutura:
<Confirmação><email do destinatário>

Seguem as seguintes confirmações da receção de notificações:

- ListenedJoinTravel <email>

- ListenedExitTravel <email>

- ListenedDeleteTravel <email>

- ListenedAddPassenger <email>

- ListenedLeaveTravel <email>

- ListenedCreateTravel <email>

5.4 Segurança

A nossa aplicação contém a tecnologia de segurança SSL no processo de autenticação. Desta forma, consegue-se criar uma ligação encriptada entre o Servidor e qualquer um dos Clientes da nossa aplicação garantido que os seus dados permanecem privados e não são comprometidos. É também utilizado um certificado que autentica as identidades e envia a informação encriptada para o Servidor.

6 Conclusão

Em virtude do que foi mencionado concluímos que a aplicação que realizamos foi bem implementada, interessante de desenvolver e de fácil utilização.

No início do desenvolvimento demonstramos dificuldades a implementar as notificações a enviar pela aplicação ao utilizador. No entanto, quando descobrimos a forma de resolver este problema, a sua implementação foi facilitada.

No futuro poderá ser feita uma interface gráfica, com o objetivo de aumentar a usabilidade. E também se poderá implementar uma forma tentar reiniciar o servidor se este tiver de ser terminado.

Os conhecimentos interiorizados no decorrer da cadeira foram fundamentais para o desenvolvimento deste projeto, pois permitiu-nos ultrapassar dificuldades na forma como deve ser feita uma aplicação cliente/servidor, com notificações, segurança e escalabilidade.

Tendo sido cumpridos todos os requisitos e implementada uma aplicação segura, escalável, tolerante a falhas e com um sistema de notificações, consideramos que o nosso trabalho foi desenvolvido para atingir a nota proposta inicialmente.