iVotas: Voto Eletrónico na UC Sistemas Distribuídos 2017/2018

Relatório Meta 1

Miguel Machado, 2014214547

Teresa Salazar, 2015234237

ÍNDICE

INTRODUÇÃO **3**

ARQUITETURA DE SOFTWARE **3**

FUNCIONAMENTO DO SERVIDOR TCP **5**

THREADS 5

PROTOCOLO 5

FALHA DOS RMIs 5

FUNCIONAMENTO DO SERVIDOR RMI **5**

MÉTODOS REMOTOS 5

CALLBACKS 7

FAILOVER 7

DISTRIBUIÇÃO DAS TAREFAS POR ELEMENTOS DO GRUPO **8**

TESTES DE SOFTWARE **9**

CONCLUSÃO **11**

INTRODUÇÃO

Este projeto tem como principal objetivo criar um sistema de voto eletrónico para eleger as direções de organismos da Universidade de Coimbra.

O voto electrónico consiste na obtenção, armazenamento e contagem de votos, por via eletrónica, relativos a um processo eleitoral. A escolha de cada votante deverá ser secreta, sendo que cada eleitor poderá votar apenas uma vez, e a contagem de votos deverá estar de acordo com as escolhas feitas pelos votantes.

ARQUITETURA DE SOFTWARE

A arquitetura do projeto é composta por diferentes componentes:

- 2 servidores RMI idênticos, um primário e outro secundário. O servidor RMI é responsável por armazenar todos os dados da aplicação, contendo vários métodos necessários ao funcionamento da aplicação.
- TCP Servers, que correspondem às mesas de voto, onde se identificam os eleitores por qualquer atributo, sendo também responsáveis por qualquer comunicação dos voting terminals com os servidores RMI.
- Admin Consoles, que permitem alterar a base de dados ou fazer determinadas configurações.
- Voting Terminals, que correspondem aos clientes TCP, permitindo aos eleitores autenticarem-se e votar.

Em relação a estruturas de dados existem diferentes tipos de identidades com atributos diferentes:

- **User** com atributos: nome, password, departamento, faculdade, contacto telefónico, morada, Cartão de cidadão e respetiva data de validade, tipo. O tipo pode ser 1, 2 ou 3, correspondendo a estudantes, docentes ou funcionários, respetivamente.
- **Departament** com atributo: nome.
- **Faculty** com atributos: nome e lista de departamentos que fazem parte dessa faculdade.
- **Election** com atributos: nome, descrição, data de início, data de fim e tipo. O tipo pode ser 1 ou 2, correspondendo a eleição para núcleo de estudantes ou eleição

para conselho geral, respetivamente.
- **CandidateList** com atributos nome, lista de membros, e tipo de membros da lista. O tipo pode ser 1, 2 ou 3, correspondendo a listas de estudantes, docentes ou funcionários, respetivamente.
- **VotingTable** com atributos id, eleição, departamento e array de terminais de voto

Relativamente à organização do código, dentro da diretoria src temos 5 diretorias:

- **Admin -** contém a classe de administração da consola, onde é feita a interação com o administrador. Esta pasta contém também a interface AdminInterface, usada para fazer um callback quando se pretende receber notificações na consola.
- **Client -** contém a classe TCPClient que corresponde ao terminal de voto.
- **Servers -** contém 2 diretorias:
    - RMI Server - contém a classe do servidor RMI e respectiva interface. Contém também a classe File Wrapper que recolhe dados dos ficheiros de objetos.
    - TCP Server - contém a classe TCPServer que corresponde à mesa de voto.
- **Data -** contém todas as classe que representam as estruturas de dados listadas anteriormente.
- (**Models -** contém a base de dados que ainda não está a ser utilizada para esta Meta.)

Na diretoria do projeto existe uma pasta de nome ObjectFiles, que contém todos os ficheiros de objetos necessários ao armazenamento de dados.

No servidor TCP, existem temos 3 tipos de threads:

- **Main -** Responsável por aceitar conexões de novos terminais de voto
- **Connection -** Responsável por lidar com a comunicação com terminal de votos
- **Menu -** Responsável por identificar utilizadores que chegam à mesa de voto, esta thread identifica um utilizador por qualquer campo e envia-os para o respetivo terminal de voto.

A nível de sockets, temos uma socket nos servidores RMI onde são trocadas mensagens UDP entre os dois servidores.

FUNCIONAMENTO DO SERVIDOR TCP

THREADS

Como foi referido anteriormente, a classe do Servidor TCP tem três tipos de threads: main, menu e connection.

A thread Main é responsável por aceitar novas conexões de terminais de voto que se queiram ligar ao servidor e criar a thread respectiva que lide com a comunicação do servidor com esse terminal de voto.

A thread Menu age como o menu da mesa de voto. Desta forma, ao identificar um utilizador na mesa de voto envia-o para um dos terminais de voto que esteja esteja livre.

A thread Connection é criada pela thread main quando se liga ao servidor um novo terminal de voto. Esta thread é assim responsável por lidar com toda a comunicação com o terminal de voto.

PROTOCOLO

O protocolo de comunicação entre os servidores e clientes tcp permite distinguir 3 tipos diferentes de operações com os seguintes formatos:

1. Pesquisa
    a. "type | search ; user | username"
2. Login
    a. "type | login ; username | iuser ; password | ipassword;
    b. "type | status ; logged | true or false
    c. "type | status ; logged | false
3. Voto
    a. "type | vote ; election | Election{name=iname-description=idescription-candidateLists= [CandidateList{name:ilistanome*usernames:userNames}]}
    b. type | vote ; username | iname ; choice | ilist or blank
    c. type | status ; vote | success
    d. type | status ; vote | failed

Quando um utilizador é identificado com sucesso na mesa de voto, a mensagem 1.a é enviada pelo servidor a um terminal de voto livre. Após a inserção da password por parte do utilizador, o terminal de voto envia ao servidor a mensagem 2.a, se a autenticação teve sucesso o servidor vai responder com a mensagem 2.b, se por outro lado não teve sucesso vai responder com a 2.c.

Quando o utilizador se autenticar com sucesso, o servidor envia para o terminal de voto a mensagem 3.a, que contém informação sobre a eleição e listas candidatas. Ao votar, o terminal de voto envia ao servidor a mensagem 3.b e que ao receber esta mensagem vai verificar se o voto é válido ou não, enviando ao terminal de voto a mensagem 3.c ou 3.d.

FALHA DOS RMIs

Sempre que o servidor faz chamada de um método remoto presente no servidor RMI, é verificado se este mesmo está disponível. Se este não estiver disponível, tenta-se ligar ao servidor RMI secundário.

Se por algum motivo ambos os servidores não estiverem disponíveis o servidor TCP vai alternando tentar ligar-se entre o principal e o secundário, até se conseguir ligar a um deles. Durante esta operação não vai haver falhas nos clientes, sendo que os que estiverem à espera de uma resposta do servidor TCP simplesmente vai-lhes aparecer a mensagem “Wait”.

FUNCIONAMENTO DO SERVIDOR RMI

MÉTODOS REMOTOS

A explicação de cada um dos métodos remotos está nos JavaDocs em anexo, no entanto, nesta secção do relatório estão algumas informações adicionais.

Quando iniciado, o servidor RMI é responsável por recolher dados da base de dados relativamente a: utilizadores, departamentos, faculdades, eleições, listas de candidatos e mesas de voto.

As consolas de administração, depois de ligadas a um dos servidores RMI, invocam diversos métodos do servidor que permitem:

1. Registar Users (estudantes, docentes, ou funcionários)

2. Criar Departamentos
3. Criar Faculdades
4. Criar Eleições
5. Criar Listas de Candidatos a uma Eleição
6. Adicionar Mesas de Voto a uma Eleição
7. Alterar propriedades de uma Eleição
8. Saber em que local votou cada eleitor
9. Consultar detalhes de eleições passadas

Quando se pretende criar um utilizador, a consola de administração invoca um método remoto do servidor para criação do mesmo, enviando depois de validados os seguintes atributos: nome, descrição, nome do departamento, nome da faculdade, contacto, morada, número e validade do cartão de cidadão. Ser existir uma faculdade e um departamento com esses nomes, é criado um novo utilizador, guardando o mesmo na base de dados.

Os métodos de criação de faculdades, departamentos, eleições e mesas de voto são idênticos ao descrito em cima.

Quando se pretende criar uma lista de candidatos é pedido inicialmente o nome da eleição e tipo. O tipo de lista pode ser 1 (eleição para núcleo de estudantes, composta apenas por estudantes) ou tipo 2 (eleição para Conselho Geral). No último caso, os membros da lista podem ser de 3 tipos: 1 (lista de estudantes), 2 (lista de docentes) ou 3 (lista de funcionários). Assim, quando são pedidos os nomes dos membros da lista, apenas podem ser adicionados utilizadores do mesmo tipo.

O método do servidor RMI para alterar propriedades de uma eleição permite ao administrador alterar o nome, descrição, data de início e data de fim da eleição. Estas funcionalidade estão devidamente protegidas, sendo que não é possível alterar a data de fim para antes da data de início assim como não é permitido alterar uma eleição que já terminou ou está a decorrer.

Sempre que se cria ou altera algum tipo de dados, o respetivo ficheiro de objetos é atualizado.

Para informar o utilizador de resultados de eleições passadas existe um método através do qual são percorridas todas as eleições e verificado se já terminaram. Em caso afirmativo, calcula-se o número de votos e percentagem de cada lista de candidatos assim como o número e percentagem de votos em branco e votos nulos.

CALLBACKS

O administrador tem a possibilidade de ver tanto o estado das mesas de votos como os eleitores em tempo real. Se o admin no menu escolher a opção de ver notificações, sempre que uma mesa de voto fica on/off é mostrado na consola o id da mesa. Igualmente, sempre que alguém vota, é mostrado o número de votos daquela eleição na consola do administrador.

Tudo isto é feito usando um callback do RMI Server para a Admin Console. Através de um subscribe, o Servidor passa a ter uma referência remota para a Consola, sendo assim possível invocar métodos remotos sobre ela.

FAILOVER

O servidor RMI, quando iniciado, começa por ser um servidor de backup, enviando de 1 em 1 segundo pings via UDP, estando à espera de novas mensagens do Servidor Primário.

Se durante 5 pings seguidos as mensagens se perderem, o servidor secundário liga-se e assume o papel de Servidor Primário, ficando também com os dados atualizados.

Se o servidor RMI original recuperar, toma o papel de secundário.

DISTRIBUIÇÃO DAS TAREFAS POR ELEMENTOS DO GRUPO

TAREFA	AUTOR(ES)
Criação de Classes	Miguel
Criação da Interface do RMI	Teresa, Miguel
Registar pessoas	Teresa
Criar Departamentos e Faculdades	Teresa
Criar Eleição	Teresa
Criar Listas de Candidatos a uma Eleição	Teresa
Adicionar mesas de voto	Teresa, Miguel
Alterar propriedades de uma eleição	Teresa
Saber em que local votou cada Eleitor	Teresa
Consolas de administração mostram mesas on/off	Teresa, Miguel
Consolas de administração atualizadas em tempo real nas eleições	Teresa, Miguel
Eleição termina corretamente nas horas	Miguel
Consultar detalhes de eleições passadas	Miguel
Avaria de um servidor RMI não tem qualquer efeito nos clientes	Teresa, Miguel

Não se perde/duplica votos se os servidores RMI falharem	Miguel
Avarias temporárias (<30s) dos 2 RMIs são invisíveis para clientes	Miguel
Terminal de voto bloqueado automaticamente após 120s sem uso	Miguel
Crash de terminal de voto é recuperado	Miguel
Heartbeats via UDP entre o servidor primário e o secundário	Teresa
Em caso de avaria longa os servidores TCP ligam ao secundário	Miguel
Servidor RMI secundário substitui o primário em caso de avaria longa	Teresa
Os dados são os mesmos em ambos os servidores RMI	Teresa, Miguel
O failover é invisível para utilizadores (não perdem a sessão)	Teresa, Miguel
O servidor original, quando recupera, torna-se secundário	Teresa
Relatório	Teresa, Miguel

TESTES DE SOFTWARE

TESTE	PASS / FAIL
Registar pessoas	
Criar Departamentos e Faculdades	

Criar Eleição	
Criar Listas de Candidatos a uma Eleição	
Adicionar mesas de voto	
Alterar propriedades de uma eleição	
Saber em que local votou cada Eleitor	
Consolas de administração mostram mesas on/off	
Consolas de administração atualizadas em tempo real nas eleições	
Eleição termina corretamente nas horas	
Consultar detalhes de eleições passadas	
Avaria de um servidor RMI não tem qualquer efeito nos clientes	
Não se perde/duplica votos se os servidores RMI falharem	
Avarias temporárias (<30s) dos 2 RMIs são invisíveis para clientes	
Terminal de voto bloqueado automaticamente após 120s sem uso	
Crash de terminal de voto é recuperado	
Heartbeats via UDP entre o servidor primário e o secundário	
Em caso de avaria longa os servidores TCP ligam ao secundário	
Servidor RMI secundário substitui o primário em caso de avaria longa	

Os dados são os mesmos em ambos os servidores RMI	
O failover é invisível para utilizadores (não perdem a sessão)	
O servidor original, quando recupera, torna-se secundário	

CONCLUSÃO

A execução deste projeto envolveu:

1. Programação de um sistema de voto eletrónico, com arquitetura cliente-servidor.
2. Sockets TCP/IP para comunicação entre clientes e servidores.
3. Modelo multithread nos servidores.
4. Camada de persistência de dados usando Java RMI.
5. Redundância com failover.

Consideramos que o nosso desempenho neste projeto correspondeu às expectativas e acreditamos que fizemos um bom trabalho.