Universidade de Coimbra

Faculdade de Ciências e Tecnologias

Departamento de Engenharia Informática

Sistemas Distribuídos

iVotas

Francisco José Rodrigues dos Santos 2015238068
Leonardo Machado Alves Vieira 2015236155
Tiago Miguel Vitorino Simões Gomes 2015238615

Outubro, 2017

Conteúdo

1 Introdução

Este projeto foi desenvolvido no contexto da disciplina de Sistemas Distribuídos e tem como objetivo a criação de um sistema de voto eletrónico para eleger as direções de vários organismos da Universidade de Coimbra. Para alcançar este fim, foi necessário desenvolver vários sistemas que comunicam entre si de modo a garantir que seja possível votar sem haver qualquer perda de dados, havendo resistência a falhas das várias partes. A comunicação é feita à base de vários protocolos (nomeadamente Java RMI, TCP e UDP) que foram usados em diferentes contextos, de acordo com as suas características, tal como será explicitado em secções seguintes.

2 Arquitetura do Sistema

Para explicarmos a arquitetura deste sistema temos que referir as seguintes entidades - Clientes TCP (Terminais de voto), Servidores TCP (Mesas de Voto), Servidores RMI, *Admin Console* e Base de Dados. Todo este sistema foi construído com base nos métodos implementados pelo servidor RMI, pois é nestes métodos que encontramos os pedidos e as respostas relativas às ações dos referidos clientes, o acesso à base de dados e ainda grande parte das proteções necessárias para garantir que os clientes não consigam realizar ações que não lhes são permitidas.

2.1 Clientes TCP

Os eleitores poderão utilizar o cliente TCP, fornecido nos materiais de apoio da disciplina, como terminal de voto. Para realizar qualquer operação, estes terão primeiro que se conectar à mesa de voto e as informações relevantes ao cliente ser-lhe-ão enviadas pela mesma ao longo de toda a sessão. Para enviar comandos, os administradores da mesa de voto terão que desbloquear o terminal e de seguida o cliente terá que escrever os comandos que deseja, seguindo as configurações de mensagem que lhe são apresentadas.

2.2 Servidores TCP

O servidor TCP contém todas as funcionalidades que o *staff* da mesa de voto necessita para poder administrar a votação local, nomeadamente, a verificação da identificação dos eleitores e a sua permissão de votar, o desbloquear de terminais e ainda todas as comunicações com o servidor RMI.

A mesa de voto começa por estabelecer uma ligação com o servidor RMI, sem a qual não iniciará. Como parte da configuração inicial, os administradores da mesa de voto terão que selecionar a eleição a que aquela mesa diz respeito e de seguida qual o ID da mesa em questão. Todas estas escolhas estarão em conformidade com o existente na base de dados. Assim que são definidas estas questões, é necessário que um dos administradores associados àquela mesa realize o login. Após tudo isto, o servidor está finalmente pronto a aceitar conexões TCP dos terminais de voto (criando um *thread* para tratar de cada novo cliente), tendo o *staff* acesso a um menu que lhes permite desbloquear terminais conectados para pessoas que passem a verificação.

Sempre que um terminal de voto envia uma mensagem, em primeiro lugar é verificado se este está desbloqueado e que a mensagem é válida (ou seja, se segue a especificação de mensagens). Basta um destes testes falhar para a mensagem ser rejeitada. Em caso de ser aceite, a mensagem é interpretada e o pedido do cliente é respondido. Se um pedido depender do servidor RMI e for momentaneamente impossível conectar com este, o servidor tenta continuamente retomar a conexão durante 30s, sem o eleitor ser notificado de algum problema. Passando 30s sem ser possível a reconexão, o pedido é cancelado e o eleitor é notificado que este não foi concretizado.

2.3 *Admin Console*

A Admin Console fornece, essencialmente, uma interface que facilita a edição de praticamente todos os dados referentes à eleição. Para isso, começa por se conectar ao servidor RMI, dando depois, ao administrador, a possibilidade de navegar vários menus e submenus, de modo a escolher a operação desejada. Todas estas operações estão associadas a métodos existentes no servidor RMI que são chamados assim que o administrador faz a sua escolha. Esta consola tem, também, a capacidade de se reconectar após uma quebra de ligação com o RMI Server.

2.4 Servidores RMI

Neste trabalho, os dois servidores RMI (principal e secundário) irão ser corridos na mesma máquina, de modo a facilitar o seu acesso pela *admin console* e pelos servidores TCP. Caso contrário, seria necessário utilizar um *proxy* para igualar os IP's das máquinas que correm os servidores, para que os restantes componentes pudessem invocar os seus métodos.

Após ter sido feito um pedido por parte do servidor TCP ou da *admin console*, esse pedido é reencaminhado, remotamente, para o servidor RMI. O pedido é então processado pelo servidor RMI, que executa a função relativa ao pedido feito. Após a execução da função, no geral, é retornado o sucesso ou o insucesso da tarefa pedido, ou em casos específicos poderá retornar outro tipo de dados.

Optámos pela opção de fazer funções especificas para cada pedido necessário, pela sua simplicidade de implementação, permissão de um código mais compreensível e rapidez no processamento de pedidos.

2.5 Bases de Dados

A base de dados *MySQL* (versao 5.7) utilizada neste projeto está implementada num sistema operativo OSX 10.13 *High Sierra* . Este sistema de dados contém toda a informação necessária à execução do iVotas.

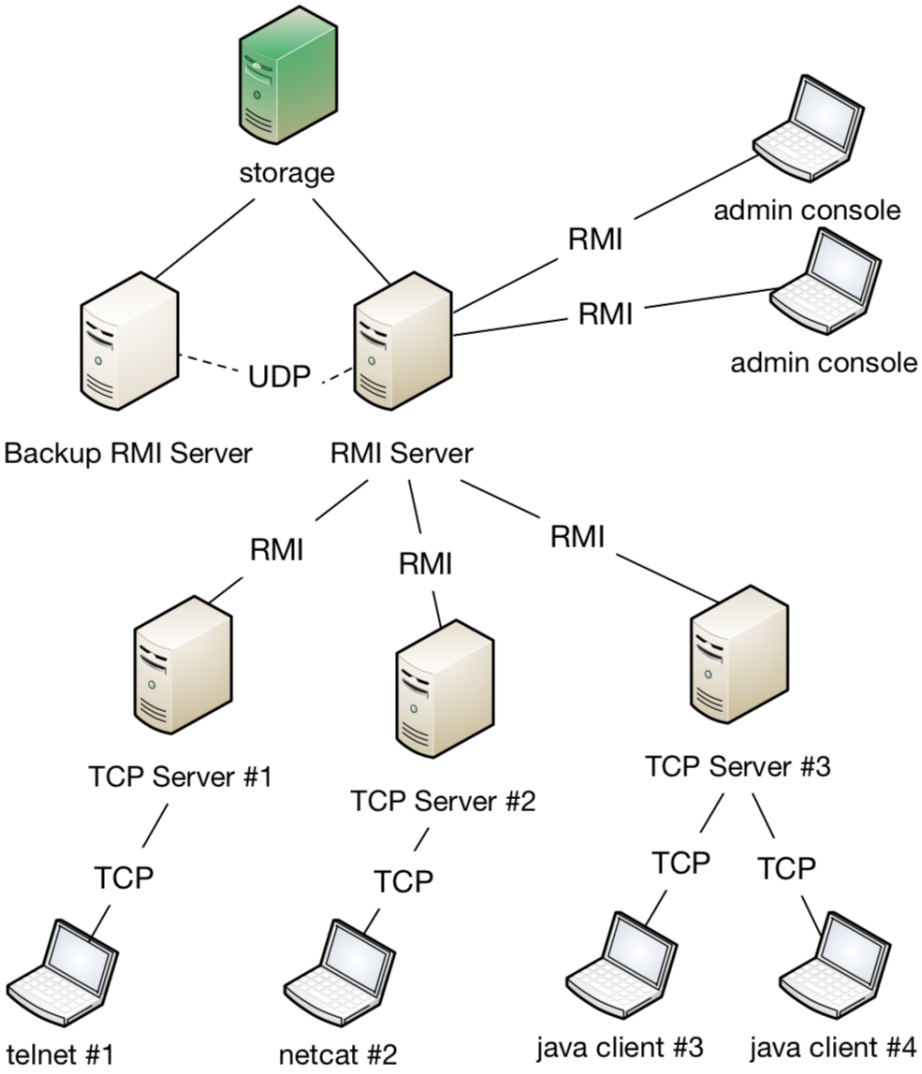

Figura 2.1: Diagrama da Arquitetura do Sistema

3 Modelo de Dados

Por forma a sustentar a informação necessária ao funcionamento do iVotas, implementámos um modelo de dados persistente através de uma base de dados *MySQL*.

Temos cerca de seis entidades que irão suportar todo o modelo de dados necessário para o projeto. No seguinte diagrama Entidade-Relação iremos apresentar todas as entidades, relações, tabelas e atributos utilizados para construir o sistema de voto eletrónico.

Para acedermos às informações provenientes das tabelas, foi utilizada uma função que submete *queries* em SQL à base de dados e retorna a resposta em ArrayList de Strings.

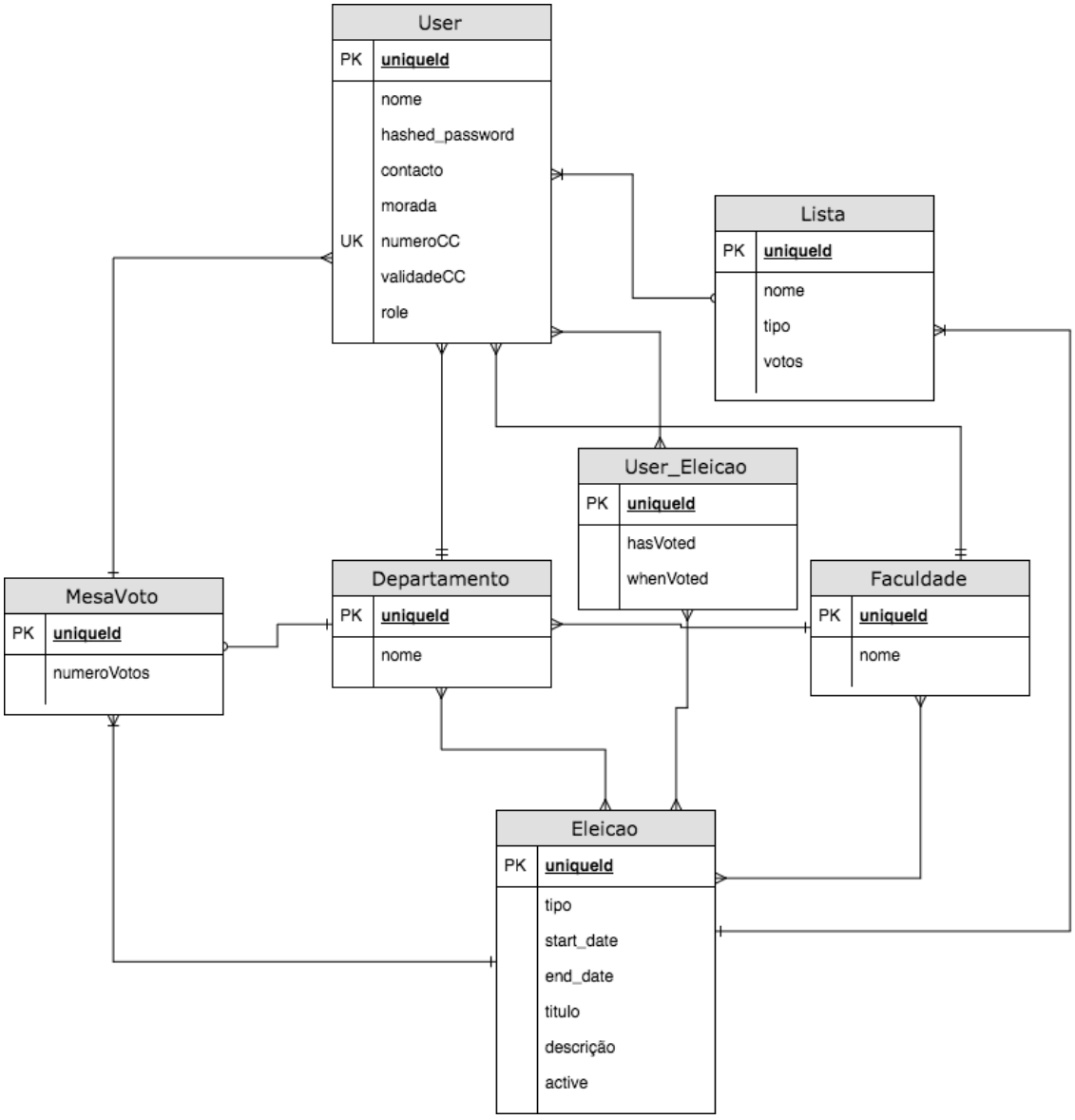

Figura 3.1: Diagrama Entidade-Relação do Modelo de Dados

4 Especificação de Protocolo

4.1 TCP

O Transmission Control Protocol é caracterizado pela sua fiabilidade no envio de dados (evita perda de dados e entregas desordenadas, entre outras) e também por ser *full-duplex*. Isto permite o seu uso em situações em que é fulcral garantir a receção de mensagens que são trocadas entre dois sistemas. Neste projeto, o protocolo é usado para todas as comunicações entre as mesas de voto e os terminais de voto, pois é essencial assegurar que não são perdidas informações tão fulcrais como um voto e que o cliente é sempre informado se os seus pedidos foram correspondidos ou não. As mensagens trocadas seguem a especificação definida pela classe Message, que permite envio de inteiros, Strings e ArrayList de Strings, tudo sob a forma de uma string. Para efeitos de demonstração, após o envio de uma mensagem da parte do TCPServer, este envia também um "Pretty Print"que simula o tratamento que uma interface gráfica do lado do cliente daria aos dados enviados (display mais intuitivo).

4.2 UDP

O User Datagram Protocol é conhecido por ser um protocolo leve e simples na transmissão de dados, que não oferece fiabilidade no que toca à troca de informação, pelo que não é indicado para troca de mensagens vitais ao sistema, pois pode haver perda de dados. Como tal, este protocolo apenas será usado entre os servidores RMI para estabelecer o sistema de pings (*heartbeat*) que possibilita a deteção da falha do servidor principal. Esta função, embora extremamente importante, tem alguma tolerância a falhas de receção/envio, pois a falha de menos de 5 pings não causa qualquer problema, sendo altamente improvável haver 5 falhas seguidas sem haver um problema real por trás destas falhas.

4.3 Java RMI

Remote Method Invocation é uma interface que permite a execução de métodos de um servidor remoto. Neste projeto, o servidor vai criar um *rmi-registry* numa porta configurável pelo utilizador e de seguida instancia objetos remotos. Estes objetos contêm referência para um nome que é *bound* a um porto. O cliente vai dar *lookup* do objeto e aceder remotamente aos seus métodos. Foi usada esta interface para implementar as seguintes funcionalidades do projeto:

- Verificar existência de um dado CC e as permissões de voto do respetivo utilizador;
- Verificar dados de login de um eleitor e membros de uma mesa de voto;
- Votar numa lista e guardar utilizadores que já votaram e onde;
- Criar / Modificar / Apagar alunos, professores, funcionários, eleições, listas, mesas, departamentos e faculdades;
- Entre outras...

5 Instalação e Configuração

5.1 TCPClient

Para correr o TCPClient, basta passar como argumento o IP da mesa de voto a conectar, seguido do porto do mesmo. Em caso de omissão destes argumentos, é assumido "localhost 12345".

5.2 TCPServer

O TCPServer necessita do ficheiro "tcp.properties"para poder executar corretamente. É nesse mesmo ficheiro que podem ser definidos o porto (tcpPort) e endereço e nome do RMIServer (rmiName). O programa deve ser corrido sem qualquer argumento adicional.

5.3 RMIServer

As configurações do RMIServer dependem dos ficheiros "rmi.properties"e "policy.all". Este último define o nível de permissões para a execução de métodos remotos, enquanto que no primeiro podem se definir o endereço e nome do RMIServer (rmiName), o porto (rmiPort), o porto UDP do servidor primário (mainUDP) e secundário (secUDP), a frequência dos pings de *heartbeat* (pingFrequency), o número de tentativas falhadas de pings até o secundário assumir o controlo principal (retries) e, por último, o endereço e porto da base de dados (dbIP e dbPort, respectivamente). O programa assume que a versão 5.7 do MySQL está instalada e não aceita argumentos adicionais.

5.4 Consola

Este programa é configurado a partir do ficheiro "adminconsole.properties". A única configuração possível é o endereço e nome do servidor RMI (rmiName). O programa deve ser corrido sem qualquer argumento.

6 Controlo de Exceções

Para evitar problemas inesperados, o controlo de exceções é um fator essencial. Todos os sistemas em questão controlam exceções de alguma forma.

6.1 Terminais de Voto

O TCPClient fornecido foi inalterado, pelo que tem apenas um controlo de exceções básico. Quando há alguma falha de conexão, o terminal de voto é encerrado corretamente, sem se reiniciar.

6.2 Mesa de Voto

Há várias exceções possíveis no que toca à mesa de voto, mas dividem-se em três grande categorias: fecho da ligação com o Servidor RMI, fecho da ligação com um terminal de voto e receção de mensagens inesperadas.

Quando há uma falha na ligação com o servidor RMI, esta é detetada assim que a mesa de voto tente fazer um pedido a esse servidor. Quando isso acontece, é automaticamente tentada a reconexão durante 30 segundos. Se ao fim desse prazo ainda não se tiver restabelecido a ligação, o pedido é cancelado, informando o cliente do sucedido, se necessário.

Quando há o fecho da ligação com um terminal de voto, o *thread* associado ao mesmo é encerrado corretamente, retirando-se da lista de terminais de voto disponíveis. A reconexão do terminal de voto será bem sucedida, assumindo a forma de um novo terminal.

Mensagens vindas de um terminal que tenham forma ou conteúdo inesperado são rejeitadas, comunicando o sucedido ao terminal que fez o envio.

6.3 *Admin Console*

A principal exceção que a consola tem que gerir são falhas de comunicação com o RMI. Assim que há uma quebra na ligação, a consola irá tentar retomar imediatamente e continuará a fazê-lo até conseguir (ou o programa ser terminado). As falhas nos métodos remotos são tratadas pelo servidor RMI e é recebida a informação referente a essa falha.

6.4 Servidores RMI

No que toca aos servidores RMI existem várias exceções importantes para o funcionamento expectável do programa.

A primeira é a falha que permite a um servidor detetar que é secundário. Esta é detetada quando o servidor de backup tentar criar um registo na mesma porta, e com o mesmo nome que o servidor principal. É desta forma que um servidor sabe que tem que assumir o papel de principal

A segunda falha importante é o *timeout* da socket utilizada pelo servidor principal, que ao fim de um segundo sem receber um ping do servidor principal incrementa um contador, que quando chegar a cinco falhas faz com que o servidor secundário assuma o papel de servidor principal

Por último, qualquer falha num método chamado remotamente é tratada e comunicada ao *caller*.

7 Failover

De forma a que haja sempre um servidor RMI considerado principal, ambos os servidores comunicam entre si, por via UDP, trocando assim *heartbeats*, que permitem que cada servidor saiba o estado de execução do outro. Caso o servidor secundário perca a conexão do principal, este vai assumir o controlo e receber as ligações dos servidores TCP e da Admin Console, sem estes necessitarem de qualquer reconfiguração.

A distinção dos dois servidores é feita aquando da iniciação do processo do servidor RMI. Caso o registo para funcionamento do servidor RMI já tenha sido criado, o servidor vai assumir que é o secundário e ficará à espera de um falha do servidor principal para assumir o controlo das operações.

O espaçamento temporal dos pings enviados é definido pelo utilizador e pode ser alterado no ficheiro de configurações do servidor RMI, bem como o IP e as portas que cada servidor irá utilizar.

8 Distribuição de tarefas

As tarefas foram distribuídas seguindo a primeira sugestão do enunciado, isto é, cada elemento ficou encarregue de uma parte, nomeadamente:

- **Francisco Santos** - RMI Server e conexões com a Base de Dados
- **Tiago Gomes** - TCP Server e conexões com TCP Clients
- **Leonardo Vieira** - Admin Console

Houve, como é normal, entreajuda em todas as partes, pelo que há contribuições de todos em todos os sistemas, para além de todos terem colaborado na criação do relatório e testes de funcionalidades.

9 Testes realizados

Tabela 9.1: Tabela de testes

Teste	**Resultado Obtido**
Registar pessoas (estudantes, docentes, ou funcionários)	Bem sucedido
Criar departamentos e faculdades	Bem sucedido
Criar eleição	Bem sucedido
Criar listas de candidatos a uma eleição	Bem sucedido
Adicionar mesas de voto a uma eleição	Bem sucedido
Identificar eleitor na mesa de voto e desbloquear terminal de voto	Bem sucedido quando o eleitor tem permissões de voto nessa eleição
Autenticação de eleitor no terminal de voto	Bem sucedido
Votar (escolher, uma só vez, uma lista no terminal de voto)	Bem sucedido
Alterar propriedades de uma eleição	Bem sucedido
Saber em que local votou cada eleitor	Bem sucedido
Consolas de administração mostram mesas de voto on/off	Lógica apenas existente server-side.
Consolas de administração atualizadas em tempo real nas eleições	Lógica apenas existente server-side.
Eleição termina corretamente na data, hora e minuto marcados	Bem sucedido
Consultar resultados detalhados de eleições passadas	Bem sucedido
Alterar dados pessoais	Bem sucedido
Gerir membros de cada mesa de voto	Bem sucedido
Considerar eleições de departamento e faculdade	Bem sucedido

Tabela 9.2: Tabela de testes

Teste	**Resultado Obtido**
Voto antecipado	Bem sucedido
Avaria de um servidor RMI não tem qualquer efeito nos clientes	Bem sucedido
Não se perde/duplica votos se os servidores RMI falharem	Bem sucedido, clientes são avisados em caso de perda
Avarias temporárias (<30s) dos 2 RMIs são invisíveis para clientes	Bem sucedido
Terminal de voto bloqueado automaticamente após 120s sem uso	Bem sucedido
Crash de terminal de voto é recuperado	Bem sucedido, basta reiniciar manualmente
Crash de um servidor TCP é recuperado	Bem sucedido, basta reiniciar manualmente
Heartbeats via UDP entre o servidor primário e o secundário	Bem sucedido
Em caso de avaria longa os servidores TCP ligam ao secundário	Bem sucedido
Servidor RMI secundário substitui o primário em caso de avaria longa	Bem sucedido
Os dados são os mesmos em ambos os servidores RMI	Bem sucedido
O failover é invisível para utilizadores (não perdem a sessão)	Bem sucedido
O servidor original, quando recupera, torna-se secundário	Bem sucedido
Servidores TCP das mesas protegidos com login e password	Bem sucedido