UNIVERSIDADE DE COIMBRA

# iVotas

## Voto Eletrónico da UC

Sistemas Distribuídos 2017/2018

# Projeto Final

**Projeto desenvolvido por:**
João Filipe Mendes Castilho nº 2014202807
João Rodrigues Martins nº2014214340

# Introdução

Foi nos proposto neste projeto, o desenvolvimento de uma simulação de um sistema de voto eletrónico. Este projeto está dividido em duas fases: a primeira, tinha como objetivo montar uma arquitetura servidor-cliente exigindo competências nas áreas de sockets TCP/IP, modelos de multithreading, Java RMI e tratamento de failovers. O objetivo da segunda fase é desenvolver uma interface Web e interligá-la com a aplicação da primeira meta  tendo em vista adquirir competências nas áreas de Struts2, JavaServer Pages, JavaBeans, arquitetura MVC, WebSockets e serviços REST.

# Arquitetura

Como é referenciado na introdução do trabalho, este projeto está dividido em duas partes. A arquitetura da 1ª parte é composta por 2 RMI's: Servidor Primário e Servidor Backup. Ambos os servidores são executados pelo mesmo ficheiro (**dataserver.jar**), no entanto, com parâmetros de entrada distintos. Estes interagem dinamicamente entre si, enviando pacotes UDP de forma a manterem o sistema ativo mesmo quando um deles falha. Para além dos servidores, na 1ª meta também foram desenvolvidos:

**Consola de administração -** executada pelo ficheiro **console.jar**

**TCPServer –** executado pelo ficheiro **server.jar**, funciona como mesa de Voto servindo como intermediário entre os terminais de Voto e o servidor RMI. Este servidor TCP é multithreading, conectando-se uma thread por cada terminal de voto. Enquanto a main thread está responsável por desbloquear os terminais de Voto ao identificar o utilizador, uma outra thread denominada por ThreadTCP está responsável pela conexão dos terminais de voto a este TCPServer.

Terminal de Voto – Terminal onde o eleitor se poderá conectar para votar na eleição desejada.

Na 2ª meta, foi desenvolvida a arquitetura Web, implementada com Struts2, JavaServer Pages e um servidor TomCat. Este WebServer também servirá como intermediário entre os eleitores/administradores e o servidor RMI. Este interpreta pedidos provenientes das páginas Web, comunica com o servidor RMI através das actions devolvendo uma resposta às páginas Web. As diferentes Actions desenvolvidas correspondem às diferentes funcionalidades disponibilizados aos eleitores ou administradores. Foi desenvolvido também uma comunicação entre a API do facebook e o servidor, usando o protocolo OAuth. Esta comunicação permite ao eleitor associar a sua conta à conta do Facebook e também permite que este se logue no servidor Web através da sua conta de

Facebook. Para além disso, também foi implementado uma comunicação real-time no browser através de WebSockets que permite visualizar alterações dos utilizadores online em tempo real.

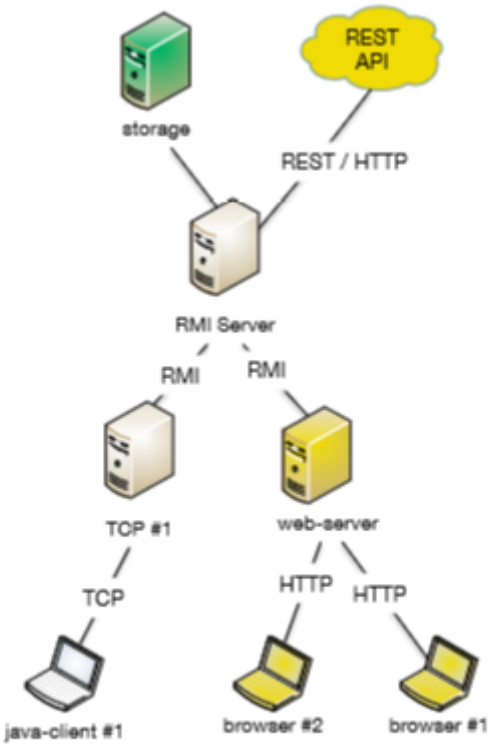


Fig. 1: Arquitetura do projecto

# Integração das Struts com o Servidor RMI

No nosso projeto, seguimos o padrão de arquitetura MVC. A página (View) submete pedidos à Action (Controller), que prepara o pedido para submeter ao Bean (Model). O Bean comunica com o Servidor RMI, executa o pedido e obtém uma resposta, resposta essa que é redirecionada para a página e assim sucessivamente. Desta forma, garantimos que tanto os utilizadores da aplicação como os utilizadores Web acedem aos mesmos dados.

# Integração dos Serviços REST com o Projeto

Através do protocolo Oauth e da library ScribeJava foram integrados os serviços da API do Facebook. É permitido ao eleitor executar 2 funcionalidades com os Serviços Rest:

**Associar conta do Facebook à conta da aplicação** - Quando um eleitor depois de estar logado na plataforma online do projeto, ao ser redirecionado para o Menu Inicial, pode associar a sua conta da aplicação à sua conta do Facebook. Ao registar os dados do Facebook, a nossa aplicação recebe o Id do Facebook do utilizador e associa esse mesmo Id à sua conta da aplicação, garantindo assim que apenas uma e só uma conta da aplicação possa estar associada à mesma conta do Facebook. Isto permite que a funcionalidade seguinte possa ser executada.

**Logar-se na aplicação com a conta do Facebook** – Se a sua conta da aplicação já estiver associada a uma conta do Facebook, o utilizador em vez de se logar com os dados da sua conta da aplicação, poderá logar-se com o Facebook clicando no botão para isso destinado.

# Implementação de Websockets

A implementação dos websockets permite a atualização de contéudo na aplicação web em tempo real.

Neste caso, através destes, o administrador tem acesso imediato aos utilizadores que estão atualmente conectados à aplicação através da web. Quando um utilizador faz login na aplicação, a página HTML, ao carregar, chama o script de JS que cria o objeto WebSocket, levando como parâmetro o URI do server. O cliente utiliza duas funções, o evento onOpen e a função Remove ( invocada no logout), onde é enviado ao servidor, através da função send, uma mensagem no formato JSON, para este adicionar ou remover o utilizador do array de users online.

Quando o servidor recebe o objeto JSON, analisa o formato da mensagem e adiciona ou remove um utilizador, consoante o tipo. Entretanto, é invocada a função “sendoToAdmin”, onde a mensagem com a atualização da lista de utilizadores é enviada aos administradores.

# Testes realizados

| | Descrição | Teste | Resultado |
|---|---|---|---|
| 1 | Login | User Existente | Sucesso |
| 2 | Login | User não existente | Falha |
| 3 | Login | Com Facebook | Sucesso |
| 4 | Registar | User não existente | Sucesso |
| 5 | Registar | User existente | Falha |
| 6 | Criar Eleição | Eleição não existente | Sucesso |
| 7 | Criar Eleição | Eleição existente | Falha |
| 8 | Criar Lista de Candidatos | Lista existente na eleição | Falha |
| 9 | Criar Lista de Candidatos | Lista não existente | Sucesso |
| 10 | Listar / consultar Eleições | | Sucesso |
| 11 | Adicionar Mesa de Voto | Mesa existente | Falha |
| 12 | Editar Utilizador | | Sucesso |
| 13 | Remove Mesa de Voto | Departamento existe | Sucesso |
| 14 | Remove Mesa de Voto | Departamento não existe | Falha |
| 15 | Adicionar Departamentos | | Sucesso |
| 16 | Remove Departamentos | Departamento existe | Sucesso |
| 17 | Remove Departamentos | Departamento não existe | Falha |
| 18 | Local de Voto | CC não reconhecido | Falha |
| 19 | Local de Voto | CC existente | Sucesso |
| 20 | Editar Eleição | | Sucesso |
| 21 | Editar Departamento | Novo nome existente | Falha |
| 21 | Editar Departamento | Novo nome não existe | Sucesso |
| 23 | Votar | Eleição disponível | Sucesso |
| 24 | Votar | Não existem Eleições | Falha |
| 25 | Término da Eleição | | Sucesso |
| 26 | Consultar eleições antigas | | Sucesso |
| 27 | Editar Utilizador | Nome já existente | Falha |
| 28 | Editar Utilizador | Novo nome não existe | Sucesso |
| 29 | Listar utilizadores online | | Sucesso |
| 30 | Associar Facebook | | Sucesso |
| 31 | Login com Facebook | | Sucesso |