eVoting: Voto Eletrónico na UC

SD 2020/21 — Meta 2 — 21 de maio de 2021 (21:59)

Resumo

O *voto eletrónico* consiste na obtenção, armazenamento e contagem de votos, por via eletrónica, relativos a um processo eleitoral. Numa eleição apresentam-se candidatos e 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 realmente feitas pelos votantes. Este projeto consiste em criar um sistema de voto eletrónico para eleger as direções de organismos da Universidade de Coimbra.

1 Objetivos do projeto

No final do projeto cada estudante deverá ter:

- Desenvolvido uma interface Web para um sistema de voto eletrónico.
- Ter integrado a interface Web com a aplicação desenvolvida na primeira meta.
- Dominado Struts2, Spring, JavaServer Pages, e JavaBeans.
- Seguido a arquitetura MVC para desenvolvimento Web.
- Aplicado Websockets para comunicar assincronamente com os clientes.
- Integrado a aplicação com serviços REST externos, usando OAuth.

2 Visão geral

Nesta segunda fase do projeto, irão criar um *"frontend"* Web para a vossa aplicação. Esta nova interface irá possibilitar que os utilizadores acedam ao vosso serviço a partir de quase qualquer dispositivo com Internet no planeta, sem necessitar de instalação de software cliente. Como a interoperabilidade é um requisito muito importante, utilizadores Web deverão aceder à mesma informação que os utilizadores na aplicação cliente-servidor da primeira meta. Para tal, o servidor Web deverá comunicar com o *RMI Server* anteriormente desenvolvido em Java RMI.

Os utilizadores deverão ter as mesmas funcionalidades, independentemente da interface que usem. Portanto a interface Web deverá listar as eleições, deverá permitir a

criação de eleições e votar. Como o aspeto interactivo é muito importante na Web, o vosso projeto deverá mostrar alterações em tempo real, nomeadamente nos votos efetuados, e através de notificações quando são criadas ou fechadas mesas de voto (da meta 1). Como os utilizadores estão cada vez mais exigentes, técnicas menos robustas como meta-refresh e iframes ocultas não serão consideradas.

Finalmente, as aplicações de hoje em dia não existem isoladas umas das outras. Através de APIs REST e OAuth, irão integrar a vossa aplicação com um serviço externo: o Facebook. Ao votar numa eleição, deve ser criado um post no facebook do autor com o link para a eleição. Deverá também ser possível partilhar no facebook um apelo ao voto numa dada eleição, publicando um link.

3 Arquitetura

A Figura 1 mostra a arquitetura geral do projeto. Os elementos a amarelo referem-se à segunda meta do projecto, enquanto os outros se referem à primeira meta. O servidor Web deverá ligar-se por RMI ao servidor de dados, garantido a interoperabilidade dos dados com os clientes da primeira meta.

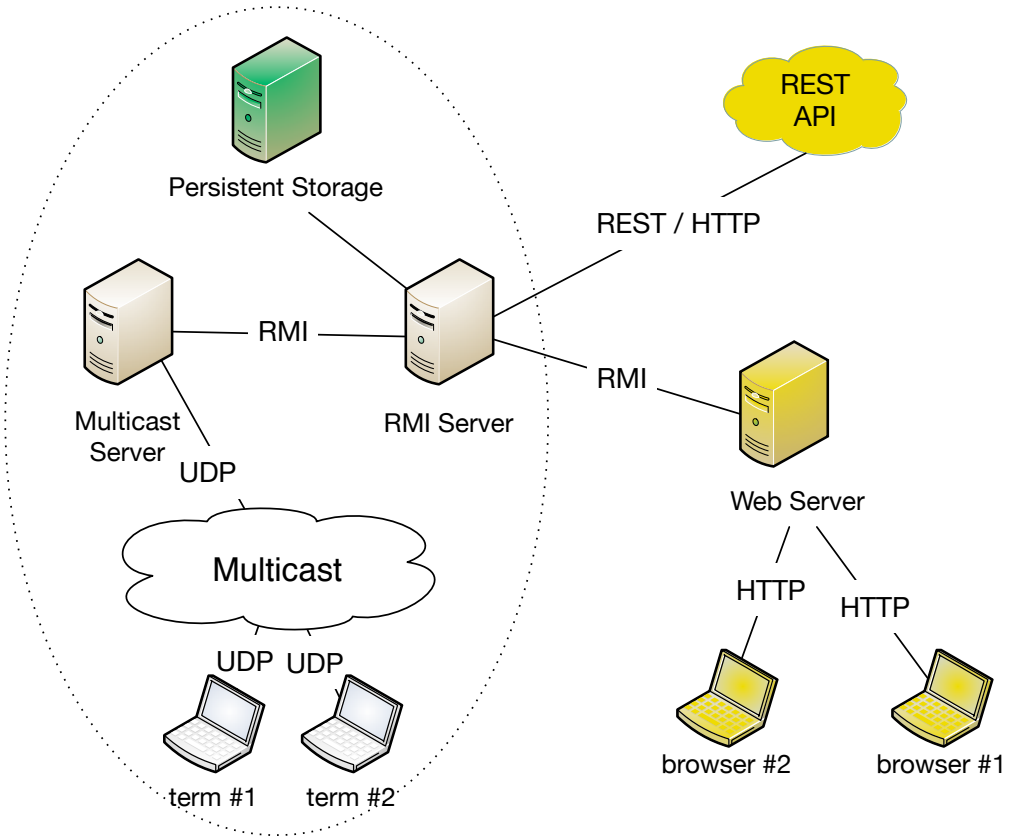

Fig. 1: Arquitetura de software do projeto.

Devem programar uma aplicação Web que corra num servidor HTTP (Apache Tomcat) e que atue como um cliente RMI para com o servidor RMI da meta 1. Os clientes irão usar browsers para se ligarem ao servidor Web para pedirem páginas através de HTTP. Para melhorar a experiência de utilização deverão ponderar fazer alguns dos pedidos via AJAX em vez de carregar a página toda.

A comunicação real-time para o browser deverá ser implementada usando WebSockets. Isto inclui notificações e alterações de valores em tempo real. Relativamente à API REST, o diagrama da Figura 1 sugere que a integração seja feita no servidor RMI, uma vez que é aí que os dados são persistidos, embora seja aceitável que possa ser feita diretamente na aplicação Web (que corre no Web server).

4 Interface Web

Pretende-se criar uma aplicação Web que disponibilize as mesmas funcionalidades da meta 1 através da Internet. Poderá optar entre criar dois portais separados (um portal para eleitores e um portal para aceder à consola de administração) ou criar um portal único no qual os eleitores e os administradores tenham usernames, passwords e permissões distintas. Usando uma arquitetura MVC, deverão programar os seguintes requisitos funcionais usando Struts2/Spring:

1. **Registar pessoas.** Deve poder-se registar estudantes, docentes e funcionários. Deverá guardar toda a informação pessoal que considere necessária, bem como uma password (código de acesso) e o departamento/faculdade ao qual a pessoa pertence. A informação pessoal deverá incluir também dados de contacto telefónico, morada, número e validade do cartão de cidadão.

2. **Criar eleição.** Uma eleição tem um momento (data, hora e minuto) em que começa e outro em que termina. Tem igualmente um título e uma descrição sucinta. Deve ser possível, de uma forma simples, restringir o grupo de pessoas que pode votar na eleição. Por exemplo, eleições para núcleo de estudantes decorrem num único departamento e podem votar apenas os estudantes desse departamento. Note que grupos de 3 alunos devem considerar eleições para conselho geral, que decorrem ao nível da universidade (em todos os departamentos), sendo que os estudantes votam apenas nas listas de estudantes, os docentes votam apenas nas listas de docentes, e os funcionários votam apenas nas listas de funcionários.

3. **Gerir listas de candidatos a uma eleição.** Uma candidatura a uma dada eleição é um conjunto ordenado de pessoas, designado por lista. Existem listas separadas de estudantes, docentes e funcionários. Uma eleição para núcleo de estudantes tem apenas como candidatas listas de estudantes, ao passo que as eleições para conselho geral têm listas de estudantes, docentes e funcionários.

4. **Gerir mesas de voto.** Deve ser possível adicionar e remover mesas de voto associadas a uma dada eleição. Uma mesa de voto é composta por uma máquina onde se identifica eleitores antes de votarem, e um ou mais terminais de voto nos quais os eleitores escolhem a lista em que querem votar. Uma mesa de voto necessita

apenas da indicação do departamento no qual está localizada (no máximo existirá uma mesa em cada departamento). Não é obrigatório que todos os departamentos tenham mesa de voto (pode haver uma mesa no DEI para servir todo o pólo 2, por exemplo). Cada mesa de voto terá o(s) seu(s) grupo(s) de multicast próprio(s).

5. **Gerir terminais de voto.** Cada mesa de voto pode ter diversos terminais de voto associados. Tal como se descreve mais à frente, a comunicação e gestão destes terminais será feita inteiramente por Multicast. Esta gestão deverá ser automática, bastando configurar o grupo de Multicast para que as máquinas se descubram e identifiquem. Uma alternativa é utilizar dois grupos de multicast distintos: um para descoberta de máquinas e outro para comunicação de votos dos terminais para o servidor da mesa.

6. **Votar.** Cada eleitor pode votar no máximo uma vez por eleição, escolhendo no terminal de voto uma das seguintes opções: uma das listas candidatas, voto em branco ou voto nulo. É fundamental que cada voto seja secreto, que cada eleitor vote apenas uma vez nas eleições em que estiver autorizado, e que todos os votos sejam contados corretamente no final. Qualquer pessoa pode votar em qualquer mesa, mesmo que seja noutro departamento que não o seu.

7. **Alterar propriedades de uma eleição.** As propriedades textuais de cada eleição devem poder ser editadas, e os instantes de início e de fim da eleição devem poder ser alterados. As alterações devem apenas poder ser feitas antes de começarem as eleições.

8. **Saber em que local votou cada eleitor.** Por uma questão de auditoria, deve ser possível saber em que mesa de voto e em que momento votou cada eleitor.

9. **Término da eleição na data, hora e minuto marcados.** No momento indicado nos detalhes de uma eleição (data, hora e minuto) o processo eleitoral termina automaticamente. Realiza-se aí o apuramento do número de votos por lista e fecha-se a possibilidade de efetuar mais votos. O resultado de uma eleição é o número de votos obtidos por cada lista candidata, os votos brancos e os votos nulos. Os detalhes dessa eleição são atualizados e podem ser consultados posteriormente.

10. **Consultar resultados detalhados de eleições passadas.** Os resultados finais de todas as eleições (que já tenham terminado) devem poder ser consultados. Estes resultados incluem o número (absoluto e em percentagem) de votos de cada lista candidata, bem como o número e percentagem de votos em branco.

11. **Voto antecipado.** Deverá ser possível votar, numa dada eleição, antes da data e hora de início oficiais. Para tal, um eleitor entra em contacto com a organização e vota através da consola de administração. **Grupos de 3 estudantes.**

12. **Alterar dados pessoais.** Deverá ser possível editar todos os dados relativos a cada pessoa (estudante, docente ou funcionário). **Grupos de 3 estudantes.**

13. **Gerir membros de cada mesa de voto.** Cada mesa de voto tem 3 membros. A composição de cada mesa deve poder ser configurada através da consola de administração. **Grupos de 3 estudantes.**

14. **Eleições para conselho geral.** As eleições para conselho geral decorrem ao nível da universidade (em todos os departamentos), sendo que os estudantes votam apenas nas listas de estudantes, os docentes votam apenas nas listas de docentes, e os funcionários votam apenas nas listas de funcionários. A aplicação deve restringir as candidaturas e votações aos perfis adequados a cada eleição. **Grupos de 3 estudantes.**

5 Notificações em Tempo Real

De forma a que a vossa aplicação seja atualizada instantaneamente (server push), deverão usar WebSockets para fazer "push" de informação para o cliente assim que esteja disponível. Deverão usar WebSockets para:

- **Página de uma eleição mostra eleitores em tempo real.** A página onde se pode consultar os detalhes de uma eleição (listas candidatas, etc.) devem também mostrar, em tempo real, o número de eleitores que votaram até ao momento em cada mesa de voto ou através da Web (numa dada eleição). Deverá mostrar-se os detalhes do número e tipo de eleitores que já votaram. Deverá diferenciar-se entre estudantes, docentes e funcionários, bem como se o voto foi realizado através da Web ou através de uma mesa de voto.

- **Páginas de administração mostram o estado das mesas de voto.** As páginas de administração recebem imediatamente (em tempo real) notificações relativamente às mesas de voto (e respetivos terminais) da meta 1 que estejam a funcionar corretamente ou não estejam em funcionamento.

- **Listagem de utilizadores ligados atualmente.** Nas páginas de administração deve ser possível visualizar a lista completa de utilizadores que se encontrem ligados à aplicação, quer através da Web, quer por se terem autenticado numa mesa de voto da meta 1.

6 Integração com serviços REST

Este projeto deverá ser integrado com o Facebook. O Facebook será usado para dinamizar a partilha social dos eleições, bem como fornecer uma alternativa ao login por username e password. Para usar a API do Facebook é necessário usar autenticação OAuth. Para tratar deste processo devem usar a biblioteca Scribejava. Não serão aceites bibliotecas que façam a integração em Java com o Facebook. As funcionalidades a programa usando REST são:

- **Associar conta ao Facebook.** Qualquer utilizador com login efetuado poderá associar a sua conta do sistema à sua conta do Facebook. Este passo permitirá depois fazer login com a conta do facebook e que os posts sejam criados automaticamente.

- **Login com o Facebook.** Um utilizador que ainda não tenha feito login, deverá poder fazê-lo com o Facebook, entrando automaticamente na sua conta sem ter de inserir username ou password.
- **Post no Facebook.** Assim que um utilizador votar, o sistema deverá automaticamente criar no Facebook um novo post indicando que o utilizador votou e fornecendo o link para a página da eleição, tudo sem interação do utilizador.
- **Partilha da página de uma eleição.** Um utilizador deverá poder fazer um post no Facebook a apelar ao voto, partilhando a página de uma eleição. Quando a eleição terminar, o mesmo post deverá ser editado e deverá passar a mostrar uma síntese dos resultados dessa mesma eleição.
- **Um administrador deverá desassociar um utilizador da conta de Facebook** Na página de administrador, deverá ser possível escolher um utilizador com Facebook associado e remover essa associação. **Grupos de 3 estudantes.**

6.1 Relatório

Devem reservar tempo para a escrita do relatório no final do projeto, tendo em conta os passos anteriores. Devem escrever o relatório de modo a que um novo colega que se junte ao grupo possa perceber a solução criada, as decisões técnicas e possa adicionar novos componentes ou modificar os que existem. **O relatório pode ser inteiramente escrito em Javadoc no código-fonte apresentado pelos estudantes.** Deve incluir:

- Detalhes sobre a integração de Struts2/Spring com o Servidor RMI da meta 1.
- Detalhes sobre a programação de WebSockets e a integração com o servidor RMI.
- Detalhes sobre a integração com o serviço REST.
- Descrição dos testes feitos à plataforma (tabela com pass/fail).

7 Entrega do projeto

O projeto deverá ser entregue num arquivo ZIP. Esse arquivo deverá conter um ficheiro README.TXT com toda a informação necessária para instalar e executar o projeto sem a presença dos alunos. Projetos sem informações suficientes, que não compilem ou não executem corretamente **não serão avaliados**.

Dentro do ficheiro ZIP deverá também estar um Javadoc/PDF/HTML com o relatório. O relatório deve seguir a estrutura fornecida, dado que a avaliação irá incidir sobre cada um dos pontos. Também no ficheiro ZIP deverá existir um ficheiro WAR com a aplicação Web pronta a executar, bem como os entregáveis da meta 1 prontos a correr.

Finalmente, o ficheiro ZIP deverá ter também **uma pasta com o código fonte completo do projeto**. A ausência deste elemento levará à anulação do projeto.

O ficheiro ZIP com o projeto deverá ser entregue na plataforma Inforestudante até ao dia **21 de maio de 2021 (21:59)**, via `https://inforestudante.uc.pt`