Stefanini Brasil - Stefa Eleição

Introdução

A ideia do projeto é criar um CRUD básico de eleições onde não é necessário de um banco de dados, pois o sistema é um simulador da urna eletrônica, para que possa cadastrar os candidatos à presidência da Stefanini.

Dicas

Antes de começar o processo do Hackathon, faça:

- Converse com os seus colegas de grupo sempre;
- Vejam o projeto todo e veja o que pode lhe ajudar para continuar na construção;
- Trabalhe em grupo sempre;
- Antes de ser sua vez, já tenha em mente o que pode ser executado, neste caso é importante que preste muita atenção nos seus colegas;
- Tem duas classe controller do JavaScript abstract.lib.js e util.lib.js que são de extrema importância para o projeto.

Arquitetura

O sistema utiliza o spring-boot, mas como não estamos utilizando banco de dados, o sistema utiliza dados em memoria utilizando objetos estáticos. O sistema utiliza um padrão de projeto MVC, onde as camadas estão divididas (Controller, BO, BancoMemoria). Exemplo:

- **Controller**
    - @RestController
    - @RequestMapping(value="/candidato")
    - public class CandidatoController {
    -
    - /**
    - * Método que adiciona candidato
    - * @param candidatoDTO
    - * @return
    - */
    - @RequestMapping(method=RequestMethod.POST, value="/criar-candidato", consumes=MediaType.APPLICATION_JSON_VALUE)
    - public List<CandidatoDTO> adicionaCandidato(@RequestBody CandidatoDTO candidatoDTO) {

- return CandidatoBO.getInstancia().insereCandidato(candidatoDTO);
- }

- **BO**
  - public class CandidatoBO {
  -
  - private SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyyMMddHHmmss");
  -
  - /**
  - * Método que gera a instância
  - * @return
  - */
  - public static CandidatoBO getInstancia(){
  - return new CandidatoBO();
  - }
  -
  - /**
  - * Método que insere candidato
  - * @param candidatoDTO
  - * @return
  - */
  - public List<CandidatoDTO> insereCandidato(CandidatoDTO candidatoDTO){
  - candidatoDTO.setIdentificador(simpleDateFormat.format(new Date()));
  - return BancoMemoriaCandidato.getInstancia().insereCandidato(candidatoDTO);
  - }
- **BancoMemoria**
  - public class BancoMemoriaCandidato {
  -
  - private static List<CandidatoDTO> candidatoDTOs = new ArrayList<CandidatoDTO>();
  -
  - /**
  - * Método que gera instância da classe BancoMemoriaCandidato
  - * @return
  - */
  - public static BancoMemoriaCandidato getInstancia(){
  - return new BancoMemoriaCandidato();

```
-         }
-
-         /**
-          * Método que insere candidatos
-          * @param candidatoDTO
-          * @return
-          */
-         public List<CandidatoDTO> insereCandidato(CandidatoDTO
    candidatoDTO){
-                 candidatoDTOs.add(candidatoDTO);
-                 return candidatoDTOs;
-         }
```

Itens que serão avaliados

- DICA: Para subir o projeto deve executar o start ou debug na classe StefaApplication;
- O sistema já possui o cadastro de candidatos, os candidatos que devem ser cadastrados:
    - Nome: Fabiane - Partido: StefaNois - Número: 101
    - Nome: Dede Programador - Partido: StefaPrograma - Número: 102
    - Nome: Roger Ágil - Partido: StefaSempreAgil - Número: 103
- A votação deve ser criados os seguintes itens:
    - A página HTML que consiga computar o voto, o campo de votos, deve conter apenas o total de votos da pessoa;
    - A controller do AngularJS, para controlar os objetos entre o HTML e o Rest. (Exemplo: cadidato.controller.js)
    - A service do AngularJS, para enviar os dados para a controladora JAVA. (Exemplo: candidato.service.js)
    - As classe de controller, bo e memória foram criados, mas necessitam que implemente os métodos que deverão executar as seguintes funções:
        - Criar o método rest do tipo POST para cadastrar o voto;
        - Criar o método rest de retorno dos votos por candidato;
        - Criar o método da camada BO;
        - Criar o processo de lógica de programação para computar o voto, exemplo:
            - CadidatoVotacaoDTO é uma classe que possui apenas a lista de candidato e seus valores;
            - VotacaoDTO é uma classe que possui o candidato e a lista de votos que é do tipo Integer.
        - Criar a lógica de retorno de candidatos e seus respectivos votos;
        - Deve se criar rotas no arquivo stefaEleicao.js.

Itens Bônus

- Atualizar os dados de um candidato a eleição, neste caso deve-se criar uma nova HTML, Controller, Service, métodos de atualização nas classe controller java, BO e BancoMemoria;
- Deletar um candidato, neste caso deve-se criar uma nova HTML, Controller, Service, métodos de atualização nas classe controller java, BO e BancoMemoria;
- Validar se um candidato já foi adicionado;
- Definir em uma tela quem ganhou a eleição de forma que o CSS deixe em caixa alta (Maiúscula).

BOA SORTE!