Projeto Vem Ser DBC
Edição 2019/02
Descrição de trabalho final

Landing Page DBC

Resumo

Seu grupo deve criar uma página para divulgação das inscrições da próxima edição do Vem Ser DBC, junto com algumas informações do projeto, visando captar candidatos para participar do processo seletivo. Esse tipo de página também é conhecida como *Landing Page*. Além da página e do conteúdo de divulgação, será necessário criar um fluxo de cadastro de candidato (currículo, informações pessoais e preenchimento de questionário) com aprovação do candidato (feita pelo perfil administrativo, do RH) e confirmação de presença na pré-seleção (feita pelo candidato).

Perfis

O sistema terá dois tipos de perfis de usuários:

- Candidato (se cadastram através do formulário)
- Administrativo (usuários pré-cadastrados pelo próprio grupo para acessar as funções administrativas, como aprovar ou rejeitar candidatos)

Lista de requisitos

- Página principal com imagens e textos para divulgação do projeto e um botão para ação principal de se inscrever no processo seletivo (conhecido como *CTA - Call To Action*).
    - Imagens podem ser solicitadas para o RH.
    - Textos: **O que é**, **Tecnologias aprendidas**, **Depoimentos**, **Benefícios**
        - Os textos devem ser criados pelo próprio grupo e passarão por revisão da staff do projeto
    - Exemplos de landing pages:
        - https://www.lyft.com/
        - https://www.airbnb.com.br/
        - https://www.spotify.com/br/
        - https://www.nubank.com.br/
    - A página deve possuir link para o site da DBC
- Cadastro de candidatos
    - Página / formulário com os seguintes campos e validações (todos campos são obrigatórios, apenas exceções serão informadas)
        - Nome completo

            - Data de nascimento (formato: dd/MM/yyyy)
            - CPF (deve estar na máscara de CPF e de acordo com o algoritmo de validação de CPF, ou seja, não pode estar apenas no formato)
                - Deve ser único, então o formulário deve validar se o CPF preenchido já existe na base.
            - Endereço (logradouro, número, complemento (opcional), cep, bairro, cidade, estado)
            - Email (validar formato de email)
                - Deve ser único, então o formulário deve validar se o email preenchido já existe na base.
            - Senha (utilizar um medidor de força de senha, exemplo: https://apertureless.github.io/vue-password-strength-meter/)
            - Confirmação de senha
            - Upload de arquivo do currículo (somente aceitar extensões pdf, deve ser salvo no servidor) exemplo: (https://www.callicoder.com/spring-boot-file-upload-download-rest-api-example/)
            - Instituição de ensino (campo livre)
        - Após o cadastro com sucesso, deve ser exibida uma página de "Estamos avaliando seu cadastro. Aguarde!" e a inscrição do candidato recebe o status "Pendente".
        - Em caso de erros no servidor, exibir devidamente na tela
- Login (perfil administrativo)
    - Deve ser implementada uma tela de login com campos usuário e senha (ambos obrigatórios)
    - A tela de login deve ser acessada a partir da página principal (deve ser colocado um link "Entrar" em algum ponto da tela)
    - Ao realizar login com sucesso, usuário deve ser redirecionado para a tela de listagem de candidatos, descrita no próximo item
- Listagem de candidatos (perfil administrativo)
    - Devem ser listados os candidatos que se cadastrarem com os seguintes campos / ações:
        - Nome completo
        - CPF
        - Email
        - Data da inscrição
        - Status inscrição (pendente, convite enviado, presença confirmada, rejeitada)
        - Enviar convite (caso pendente, passa para "Convite enviado")
            - Enviar email automático para candidato, descrito em "Enviar email convite") exemplo: (https://www.baeldung.com/spring-email - utilizar apenas as referências para Spring Boot)
        - Reenviar convite (caso "Convite enviado")
            - Reenviar email automático para candidato, descrito em "Enviar email convite")

        - Rejeitar (caso pendente, passa para “Rejeitada”)
            - Enviar email automático para candidato, descrito em “Enviar email rejeição”)
        - Link para baixar CV (em pdf)
- Detalhe de inscrição
    - Ao clicar na linha da inscrição (na listagem de candidatos), o sistema deve abrir uma tela detalhando todas informações que foram cadastradas na inscrição.
- Enviar email convite
    - O sistema deve enviar um email automático (serviço gratuito deve ser escolhido por alunos em concordância com instrutores) para o candidato, informando que a inscrição foi aprovada, mas que o mesmo deverá confirmar a presença na pré-seleção clicando em um link no email.
    - Ao abrir o link, o sistema deverá alterar o status da inscrição para “Presença confirmada”
    - O template do email, que poderá ser um HTML, pode ser definido pelo grupo. Mas, obrigatoriamente, deve ser exibido o nome do candidato no email e os links devem ser únicos por candidato.
- Enviar email rejeição
    - O sistema deve enviar um email automático (serviço gratuito deve ser escolhido por alunos em concordância com instrutores) para o candidato, agradecendo pela participação, mas informando que a inscrição foi rejeitada.
- Validar CEP
    - Utilizar o WebService gratuito para validação de CEP: https://viacep.com.br/
    - Preencher os campos Rua, Bairro, Cidade, Estado automaticamente com o retorno do webservice, quando possível

Observações gerais

1. Frontend deve ser desenvolvido utilizando React (componentizado)
2. Backend deve ser desenvolvido utilizando Spring Boot, com RestController, Service, Repository, Entity, respeitando a responsabilidade de cada classe. Deve ser utilizada autenticação OAuth2 com Token JWT. (exemplo de como recuperar o user: https://www.baeldung.com/get-user-in-spring-security)
3. Os usuários não podem ter acesso a dados de outros usuários, nem mesmo forçando por requisições no postman, a autenticação tem que validar os dados enviados com os dados de autenticação.
4. A aplicação deverá ser publicada no Heroku (guide: https://devcenter.heroku.com/articles/deploying-spring-boot-apps-to-heroku)
5. Deve ser criado um repositório privado no bitbucket para o grupo e compartilhado acesso com todos facilitadores (liberar acesso para **marcos_invox**).
6. A data limite para commits do grupo no repositório é **23:59:59 do dia 25/09/2019**
7. Todos integrantes devem trabalhar em todas as áreas de conhecimento envolvidas. Grupos que deixarem atividades específicas para cada integrante terão a avaliação

comprometida (ex: ciclano trabalha apenas em CSS ou fulano trabalha apenas em modelagem de banco).

Cronograma

- 16/09: início dos trabalhos
- 17/09 até 25/09: desenvolvimento
- 26/09: apresentações dos resultados para colegas, diretoria e staff
- 27/09: banca técnica de avaliação

Checklists

Para ajudar na avaliação dos trabalhos, utilizaremos dois checklists: um técnico e outro geral. Seguem os itens a serem avaliados e seus respectivos pesos.

Checklist técnico

Item	Peso
Cobertura de testes unitários	1
Complexidade de código (duplicação, princípios SOLID, performance, etc)	1,25
Legibilidade do código (nomes de variáveis, nomes de métodos, comentários úteis)	0,5
Estrutura do código React (componentização e boas práticas JavaScript)	2,5
Estrutura do código Spring (boas práticas da arquitetura, RestController, Service, Repository, Entity, Pagination)	2,5
Modelagem de banco	1,5
Qualidade do HTML (legibilidade e semântica)	0,25
Qualidade do CSS (sem duplicação e com seletores otimizados)	0,25
Performance na utilização dos recursos web (CSS e JavaScripts desnecessários, imagens muito pesadas, requests HTTP muito pesadas)	0,25

Checklist geral

Item	Peso
Funcionamento do software (quantidade de bugs, funcionalidades implementadas)	5
Trabalho em equipe (organização, comunicação, planejamento)	3
Engajamento dos participantes da equipe	2

Links úteis

https://trello.com/ - Micro gerenciador de tarefas
https://hangouts.google.com/webchat/start - Conferências em equipe
https://balsamiq.com/ - Mockups e prototipação
https://proto.io/ - Mockups e prototipação
https://coverr.co/ - Vídeos gratuitos para utilizar de fundo

Dicas

- Antes de commitar seu código, teste ele bem
- Após baixar o código dos colegas, teste a aplicação antes de começar a trabalhar (para evitar começar a trabalhar com problemas gerados por outros códigos)
- Estabeleçam metas diárias para concluir as coisas
- Não deixem para commitar as coisas muito em cima do deadline (limite de horário). Tentem estabelecer um horário de congelamento de código (ex: no último dia, após 12hs, ninguém mais commita, apenas se testa e faz correções urgentes)