Projeto Vem Ser DBC

Edição 2019/02

Descrição de trabalho final

**Feedback DBC**

**Resumo**

Seu grupo deve criar um sistema de feedbacks para que os gestores possam dar os feedbacks para os profissionais da DBC. É fundamental para a evolução dos profissionais receber feedbacks e quais pontos melhorar. Deve ser possível listar os feedbacks, onde terá um filtro por projeto e por gestor. Além disso teremos um fluxo de acompanhamento dos feedbacks, onde o gestor poderá dar o feedback e quais pontos melhorar, o funcionário pode responder para pedir mais informações, ou para esclarecer algo, também terá uma lista de pontos de melhoria que o gestor pode colocar e ir dando "Ok" conforme forem feitos.

**Perfis**

O sistema terá dois tipos de usuário, com permissões diferentes:

* Funcionário (pessoa que recebe feedbacks)
* Gestores (pessoa que tem uma equipe e seta feedbacks)

**Lista de requisitos**

* Página inicial
* A página inicial do sistema, caso o usuário não esteja logado, é a de login (item 3), com um link embaixo “Não possuo cadastro”, que abrirá a tela de cadastro de usuários (item 2)
* Se estiver logado tem que aparecer mensagem de boas vindas, Listagem dos feedbacks.
* Cadastro de usuários
* Página / formulário com os seguintes campos e validações (todos campos são obrigatórios, apenas exceções serão informadas)
* Nome completo
* Email (validar formato de email, aceitará apenas emails que sejam @dbccompany.com.br)
* 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://www.npmjs.com/package/react-password-strength>)
* Confirmação de senha
* Upload de foto de perfil (deve ser salvo no servidor) tem que ser opcional. Exemplo: (<https://www.callicoder.com/spring-boot-file-upload-download-rest-api-example/>)
* Após o cadastro com sucesso, usuário já deve cair logado na página inicial.
* Login (design OK)
* Deve ser implementada uma tela de login com campos usuário e senha (ambos obrigatórios)
* A tela de login deve ser acessada sempre que um usuário não estiver com uma sessão / token válido e tentar acessar alguma página ( devemos salvar no localStorage o token )
* Ao realizar login com sucesso, usuário deve ser redirecionado para a tela inicial.
* Listagem de feedbacks
* Devem ser listados os feedbacks com os seguintes dados:
* Projeto
* Nome do gestor
* Cliente
* Cargo do profissional no projeto
* Agora - última data de alteração (ex: há 5 dias, há 1 minuto, há 2 horas)
* A ordem de exibição dos feedbacks deve ser por data de atualização. Ou seja, os feedbacks que foram atualizadas por último devem vir antes.
* Ao clicar em um feedbacks, a tela de detalhamento deve ser aberta (item 5).
* Detalhe dos feedbacks
* Criar página com os seguintes campos:
* Projeto
* Nome do gestor
* Cliente
* Cargo
* Feedback do gestor
* Listagem dos pontos de melhoria com checkbox
* Campo para mensagem entre gestor e funcionário
* Caso o perfil do usuário seja de gestor é possível adicionar feedback (item 7).
* Listagem de projetos
* Devem ser listados os projetos com os seguintes campos:
* Nome do projeto
* Cliente
* Gestor
* Perfis de gestor podem criar projetos, editar e inativar (item 8)
* Cadastro de feedbacks
* Gestores devem poder adicionar feedbacks com os seguintes campos:
* Seleção de projeto (somente deve aparecer os projetos em que o usuário logado é gestor).
* Seleção de funcionário (deve aparecer todos os usuários que tem perfil funcionário \*\*\*confirmar, mas acho que um só funcionário por vez, certo?\*\*\*
* Campo para feedback
* Caixa de adição dinâmica de checkbox com pontos de melhoria.
* Após um feedback cadastrado, o gestor pode incluir mensagens novas e marcar como concluído os pontos de melhoria ou adicionar novos.
* Criar Projeto
* Perfis de gestor devem poder adicionar, editar ou inativar projetos com os seguintes campos:
* Nome do projeto
* Cliente
* Gestor (seleção de usuário tipo gestor)
* Gamification
* Deve ser possível criar categorias Badge
* Deve ser possível fazer upload da imagem do badge (<https://www.callicoder.com/spring-boot-file-upload-download-rest-api-example/>)
* Nome do Badge
* Os gestores podem dar as badges para os funcionários, tem que aparecer no topo da listagem de feedbacks

**Observações gerais**

* Frontend deve ser desenvolvido utilizando React (componentizado)
* 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> )
* 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.
* A aplicação deverá ser publicada no Heroku (guide: [https://devcenter.heroku.com/articles/deploying-spring-boot-apps-to-he HYPERLINK "https://devcenter.heroku.com/articles/deploying-spring-boot-apps-to-heroku"roku](https://devcenter.heroku.com/articles/deploying-spring-boot-apps-to-heroku) )
* Deve ser criado um repositório privado no bitbucket para o grupo e compartilhado acesso com todos facilitadores (liberar acesso para **marcos\_invox**).
* A data limite para commits do grupo no repositório é **23:59:59 do dia 25/09/2019**
* 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)