**Teste FullStack Java e ReactJS**

1. Funcionalidades e requisitos:
   1. Cadastro de livros
      1. O backend deve expor uma rota para realização do cadastro.
      2. O frontend deve ter um formulário para realizar o cadastro.
   2. Listagem de livros
      1. O backend deve expor uma rota para a listagem, deve aceitar alguns filtros (autor, ano, gênero por exemplo) e parâmetros da paginação (tamanho da página, qual página deve mostrar).
      2. O frontend deve mostrar a lista paginada, além das opções de filtragem.
   3. Detalhamento de um livro
      1. O backend deve expor uma rota para buscar todos os dados de um livro, utilizando um identificador único para o mesmo.
      2. O frontend deve ter uma página que mostre os dados de forma simples.
   4. Alugar um livro
      1. O backend deve expor uma rota para mudar o status do livro para alugado. Um livro alugado não pode ser alugado novamente.
      2. O frontend deve possibilitar a ação de alugar um livro. Caso esteja alugado, a ação deve ser de devolução.
   5. Excluir um livro
      1. O backend deve expor uma rota para exclusão de um livro. Um livro só pode ser excluído caso não esteja alugado.
      2. O frontend deve possibilitar a ação de exclusão somente quando o livro não estiver alugado.
   6. Cadastro de usuário e login
      1. Nenhuma outra funcionalidade é acessada sem estar autenticado.
      2. Deve ser utilizado token JWT para autenticação.
2. Premissas:
   1. a. O sistema deve ser desenvolvido com Java no backend e ReactJS no frontend, utilizando Typescript;
   2. O front-end deve ser iniciado com vite e não CRA (Create React App -> Depreciado).
   3. A versão do react deve ser 18 ou superior
   4. A versão do Java deve ser 17 LTS ou superior
   5. Utilizar Spring Boot
   6. Utilizar nome de rotas e métodos HTTP adequadamente
   7. O front-end pode utilizar algum gerenciamento de estados (Zustand, Redux)
   8. Ter conhecimento em tailwind, material ui ou bootstrap.
   9. O front-end deve ter uma área com rotas autenticadas para realização das ações
   10. Ter conhecimento em UX/UI para um design consistente. (Design responsivo é essencial)
   11. Front-end deve ser componentizado e organizado
   12. Os dois lados do sistema devem conversar para a realização das ações do usuário;
   13. O backend deve ser uma API REST;
   14. O frontend não precisa ser uma obra de arte, mas consistência é essencial;
   15. O código deve utilizar algum tipo de Linter para verificação de code smells;
   16. f. O código deve ser entregue em algum repositório Git público, como GitHub, GitLab ou semelhantes;
   17. O banco de dados utilizado pode ser MongoDB ou PostgresSQL, a critério do candidato;
   18. Testes automatizados (unitários, integração) são bem-vindos;
3. Critérios de avaliação:
   1. Funcionalidades desenvolvidas;
   2. Testes realizados;
   3. Uso do Git;
   4. Validação de regras de negócios e requisitos;
   5. Estruturação, formatação e legibilidade do código;
4. Diferenciais:
   1. Uso de Docker para deploy do sistema;
   2. OpenAPI / Swagger como forma de documentação da API;
   3. c. Uso de índices no banco de dados para busca efetiva;
5. Sobre a realização da prova:
   1. O candidato tem 5 dias para desenvolver o sistema;
   2. b. Caso não seja possível implementar tudo, o código deve ser entregue para a avaliação da mesma forma, comentando sobre o que foi desenvolvido, o que deixou de ser desenvolvido e as dificuldades encontradas no processo.