Teste de recrutamento DTI

Gustavo Costa Tayer

Dado a prova e os problemas sugeridos optei pela seguintes decisões:

- Utilizar um banco de dados muito utilizado no mercado (Postgres)
    - Para funcionar sem maiores problemas em outras máquina, foi criada uma instância em um docker-compose contendo a imagem do banco de dados;
        - $ docker-compose up
- Tecnologia de backend escolhida foi o NestJs:
    - É um framework para construir aplicações de server-side utilizando Node.js de forma eficiente e escalável;
    - Possui uma arquitetura bem padronizada, de forma a manter um projeto organizado e escalável;
    - Possui os chamados Blocks que é um design pattern muito utilizado;
    - Possui injeção de dependência
    - Possui excelentes ferramentas de linha de comando (CLI)
    - Fácil integração com bancos de dados e outras bibliotecas
    - Para compilar:
        - $ cd server && npm install && npm start
- Para facilitar a documentação da API optou-se pelo Swagger:
    - Para acessá-lo: http://localhost:3003/swagger
    - Para gerar e baixar o json: http://localhost:3003/swagger-json
- Testes unitários utilizando Jest:
    - dentro da pasta server rodar o seguinte comando:
        - $ npm run test:cov
- Tecnologia de frontend, como requisitado: ReactJs:
    - Para entregar algo elegante e dentro do prazo de entrega decidiu-se utilizar a biblioteca de componentes Material-ui
    - Redux foi estruturado com os seguintes dados iniciais:
        - Estoque:
            - list: uma lista com todos ids encontrados na busca
            - map: Objeto { [estoeque.id]: estoque}
                - Dessa forma, quando é preciso encontrar um estoque específico dentro do redux, o caminho é direto, não sendo mais necessário ficar buscando dentro de um Array.
            - Loading && createLoading
                - Variáveis para controlar os componentes de carregamento
    - Para compilar:
        - $ cd app && npm install && npm start