Processo Selectivo Desenvolvedor - HOPE

O processo seletivo em questão, para a posição de desenvolvedor no Grupo Hope Lingerie, requer habilidades em PHP, Docker, ReactJS e VTEX IO, com o objetivo de desenvolver e manter as lojas responsáveis.

Prova Prática

O processo seletivo para a vaga de desenvolvedor no Grupo Hope Lingerie terá duração de 7 dias e terá como objetivo a produção de um relatório que cumpra uma série de requisitos:

- Utilização de um container Docker.
- Criação do banco de dados no container Docker.
- Desenvolvimento de uma API em PHP comunicação com o banco e com o Front-end.
- Desenvolvimento de uma solução de autenticação API - Front-End.
- Utilização de ReactJS para criação de uma interface, segura e autenticada para listagem dos dados do banco.

É de suma importância a utilização de todas as tecnologias a seguir:

- PHP 7+
- Docker
- Banco de dados MySQL.
- React JS
- TypeScript

Passo 1 - Git

1. Inicializar um repositório no Git.
2. Subir o repositório no GitHub, como público.
3. Criar uma pasta api, e uma pasta web, para o projeto back-end e para o front-end, respectivamente.
4. Preferencialmente usar commits pequenos, com nome da feature/bugfix/release feita.

Passo 2 - Docker

1. Instale o Docker em sua máquina (se ainda não estiver instalado);
2. Crie um diretório "hope-teste" em sua máquina;
3. Mova os arquivos do zip para essa pasta;
4. Faça o build do projeto;
5. Abra um navegador da web e navegue para "http://localhost". Você deve ver o PHP-Apache funcionando corretamente.

Passo 3 - PHP

1. Crie um database no mysql Docker com nome “hopeDatawarehouse”;
2. Crie uma tabela com nome “hopeData” e preencha os dados com o JSON dentro da pasta JSON;

3. Depois do banco de dados, é necessário fazer a comunicação com o banco usando PHP, o código já está pronto, basta alterar a autenticação colocando os dados de conexão criados no Docker;
4. Agora faça um SELECT para retornar todos os dados e monte uma API para retornar esses mesmos dados no formato de JSON usando PHP.

Passo 4 - React JS

1. Inicialize um APP React com TypeScript (CRA ou Vite).
2. Limpe todos os arquivos desnecessários na instalação.
3. Desenvolva uma solução de estilização e crie 2 páginas:
    1. /index: página de Login com autenticação e salvamento de sessão.
    2. /dashboard: página de listagem de dados do banco.
4. Desenvolva uma solução de autenticação JWT, onde a sessão do usuário é salva, e é necessário fazer login com e-mail e senha. Os dados de autenticação devem ser salvos num Contexto (Context API) ou utilizando uma store (REDUX).
5. A listagem deve ser buscável e ordenável.

> Serão analisados: tamanhos dos commits, lógica de programação, estilização e código limpo.

Prova Teórica

1. De acordo com a prova prática, qual é a finalidade do arquivo `Dockerfile`?

    a. Configurar o ambiente do PHP-Apache.

    b. Configurar o ambiente do MySQL.

    c. Configurar o ambiente do Docker.

    d. Nenhuma das anteriores.

2. De acordo com a prova prática, qual é a finalidade do arquivo `docker-compose.yml`?

    a. Configurar o ambiente do PHP-Apache.

    b. Configurar o ambiente do MySQL.

    c. Configurar o ambiente do Docker.

    d. Nenhuma das anteriores.

3. Como o Docker Compose sabe que precisa criar dois contêineres diferentes para o PHP-Apache e o MySQL?

    a. Por meio dos nomes dos serviços definidos no arquivo `docker-compose.yml`

    b. Por meio dos nomes dos contêineres definidos no arquivo `docker-compose.yml`

    c. Por meio dos nomes dos diretórios em que o arquivo `docker-compose.yml` está contido

    d. Nenhuma das anteriores

4. Qual é a finalidade da seção `depends_on` no arquivo `docker-compose.yml`?

   a. Informar ao Docker Compose qual contêiner deve ser iniciado primeiro

   b. Informar ao Docker Compose qual contêiner deve ser iniciado por último

   c. Informar ao Docker Compose qual contêiner depende de outro contêiner

   d. Nenhuma das anteriores

5. Qual é a finalidade do comando `docker-compose up`?

   a. Criar um contêiner Docker

   b. Iniciar um contêiner Docker

   c. Criar e iniciar vários contêineres Docker em um único comando.

   d. Criar e iniciar vários contêineres Docker em um único comando, com base nas instruções do arquivo "docker-compose.yml".

6. Qual é a diferença entre `useState()` e `useReducer()`?

   a. `useState()` e `useReducer()` são funcionalidades distintas do React para gerenciamento de eventos.

   b. `useReducer()` é utilizado apenas para gerenciamento de estados complexos e `useState()` para estados simples.

   c. `useState()` é mais poderoso e flexível do que `useReducer()`.

   d. `useReducer()` é adequado para componentes com poucos estados e `useState()` é melhor para componentes com muitos estados.

7. O que são refs em React e quando são usadas?

   a. Refs são utilizadas para definir o estado inicial de um componente React.

   b. Refs são utilizadas para criar referências a elementos HTML em componentes React e nunca devem ser utilizadas para outras finalidades.

   c. Refs são utilizadas para criar componentes dinâmicos em React.

   d. Refs são utilizadas para acessar elementos do DOM e controlar o seu comportamento a partir de componentes e para armazenar valores que permanecerem pelas renderizações.

8. O que é estado ?

   a. Estado em React é a capacidade de armazenar dados dentro de um componente e gerenciá-los através de props.

   b. Estado em React é a representação visual do componente na tela.

   c. Estado em React é a camada de abstração que permite ao desenvolvedor se comunicar com o back-end para atualizar na tela.

d. Estado em React é a capacidade de armazenar dados dentro de um componente e gerenciá-los através de funções específicas, como setState.

9. O que é Redux? Como se compara com a Context API?

   a. O Redux é uma biblioteca JavaScript criada especificamente para o gerenciamento de estados em aplicações React, já a Context API é nativa do React.

   b. A Context API é uma solução de gerenciamento de estado mais performática do que o Redux, pois reduz a quantidade de renderizações necessárias na aplicação.

   c. O Redux é uma solução de gerenciamento de estado mais escalável do que a Context API, sendo capaz de gerenciar estados em aplicações de qualquer tamanho.

   d. A Context API é mais adequada para situações em que os estados precisam ser compartilhados entre vários componentes, enquanto o Redux é mais apropriado para gerenciamento de estados em componentes individuais.

10. Qual a diferença entre o método `componentDidMount()` e o hook `useEffect()` no React?

    a. Não há diferença, ambos são utilizados para executar código após a renderização do componente.

    b. O `componentDidMount()` é utilizado apenas em componentes de classe, enquanto o useEffect() é utilizado apenas em componentes funcionais.

    c. O método `componentDidMount()` é mais performático que o hook `useEffect()`, pois é otimizado para atualizações de estado mais simples.

    d. O método `componentDidMount()` é executado apenas uma vez, enquanto o hook `useEffect()` é executado sempre que há uma atualização no estado ou nas propriedades do componente.