**RELATÓRIO VITRINE DE TALENTOS – ULTRACAR**

Desenvolvedor responsável: Kelder Mendonça Passos

**Para rodar a aplicação:**

Faça o clone do repositório, acesse a pasta **/desafio-ultracar/front-end/** pelo terminal e rode o comando **npm run server** para inicializar o banco de dados e em outra janela de terminal o comando **npm run dev** para iniciar a aplicação.

**Estratégia:**

O desafio trata-se da construção de um fluxo de registro de um serviço que envolve a identificação do cliente dentro do sistema, a configuração do serviço em si (verificação dos dados, designação de um profissional, necessidade de peças extras) e confirmação final do processo.

Para tal, desenvolvi três telas correspondentes a cada etapa primordial utilizando **React** junto ao **Next.js** na linguagem **Typescript** e **Tailwind CSS** para estilização. As escolhas das tecnologias não foram à toa: **React** é o cerne do desafio e da tecnologia de front-end da Ultracar, **Next.js** tem se apresentado como um grande framework de **React**, pois facilita a criação de rotas e permite a execução da aplicação no formato **SSR** (Server-Side Rendering) e **Typescript** provê segurança e produtividade na hora do desenvolvimento de aplicações web devido ser fortemente tipada. **Tailwind CSS** por sua vez tem se tornado o framework de **CSS** padrão do **Next.js** e corresponde bem ao tamanho da aplicação.

A tela inicial conta com um simples formulário, onde um colaborador pode adicionar um dado único de um cliente, escolhi o CPF, e verifica se ele existe no sistema. Para emular um banco de dados, utilizei o **Json Server** que simula o comportamento de uma api e os verbos http. Quando o usuário é encontrado, a aplicação envia seus dados vindos da requisição para um contexto global, feito com **Context Api** e **useReducer** e redireciona a página para próxima rota.

Na segunda página, o **QR Code** é gerado através de uma biblioteca e fornece as informações recuperadas da página anterior via contexto. O usuário da página pode designar o colaborador que será responsável pelo serviço e adicionar se haverá a necessidade de peças extras para fazê-lo. Essas informações novas são adicionadas ao contexto global, junto as do cliente, para que sejam resgatadas na última tela.

A tela final é a de confirmação do pedido que recebe todas as informações acumuladas no fluxo completo, assim como uma data gerada automaticamente pelo sistema.

**Pontos fortes:**

- Desenvolvimento em **Typescript** e **Next.js**;
- O fluxo corresponde bem ao que foi pedido;
- Utilização de um contexto global e **reducer** evitando **prop drilling**;
- Utilização do **Json Server** para simular uma api;
- Um bom sistema de interfaces e tipagem;

**Pontos fracos:**

- Ausência de testes unitários;
- Falta de validações e tratamento de erros para situações como rotas não encontradas ou informações não existentes no banco de dados simulado;
- Não foram cumpridos os pontos importantes a seguir:
  - listar em quais serviços a colaboradora Joana se encontra;
  - criação de uma tela para exibição de todos os serviços em andamento;

Gostaria de destacar que possuo todas as habilidades necessárias para desenvolver os pontos fracos acima, porém não consegui demonstrá-las devido à falta de tempo suficiente e porque preferi priorizar fazer o essencial da melhor maneira que eu pude. Além disso, também sou capaz de transformar essa aplicação em **Full-Stack** com contêineres **Docker**.

Obrigado pela oportunidade!