Desafio Técnico: Sistema de Reserva de Automóveis

Objetivo:

Desenvolver uma aplicação web que permita a criação, listagem e pesquisa de reservas de automóveis. O objetivo é avaliar as competências técnicas em desenvolvimento backend e/ou frontend, assim como a capacidade de design e organização do código.

Requisitos do Backend (Java + Spring Framework):

1. API RESTful:

- Entidades:
  - Carro:
    - ID (UUID)
    - Modelo (String)
    - Marca (String)
    - Ano (Integer)
    - Disponível (Boolean)
  - Cliente:
    - ID (UUID)
    - Nome (String)
    - Email (String)
    - Telefone (String)
  - Reserva:
    - ID (UUID)
    - ClienteID (UUID - Relacionamento com Cliente)
    - CarroID (UUID - Relacionamento com Carro)
    - Data de Início (LocalDate)
    - Data de Fim (LocalDate)
- Endpoints:
  - GET /carros - Listar todos os carros.
  - POST /carros - Adicionar um novo carro.
  - GET /clientes - Listar todos os clientes.
  - POST /clientes - Adicionar um novo cliente.
  - GET /reservas - Listar todas as reservas.
  - POST /reservas - Criar uma nova reserva.
  - GET /reservas/{id} - Buscar uma reserva por ID.

2. Validações:
- O carro deve estar disponível para ser reservado.
- A data de início deve ser anterior à data de fim.
- O cliente deve existir no sistema.

3. Banco de Dados:

- Usar um banco de dados em memória (H2) para simplificar a configuração.

Requisitos do Frontend (React + Next.js):

1. Páginas:
- Página Inicial:
  - Exibe a lista de carros disponíveis.
  - Permite criar novos carros.
- Página de Clientes:
  - Lista todos os clientes.
  - Permite adicionar novos clientes.
- Página de Reservas:
  - Lista todas as reservas existentes.
  - Permite criar novas reservas.

2. Funcionalidades:
- Formulário para criar um carro (Modelo, Marca, Ano, Disponível).
- Formulário para criar um cliente (Nome, Email, Telefone).
- Formulário para criar uma reserva (Cliente, Carro, Data de Início, Data de Fim).

3. Estilização:
- Usar CSS ou bibliotecas como Tailwind ou Styled Components (opcional).

4. Consumo de API:
- Realizar chamadas para a API desenvolvida no backend para exibir e salvar dados.

Critérios de Avaliação:

1. Backend:
- Organização e clareza do código.
- Implementação correta dos endpoints e validações.
- Documentação básica (Swagger ou Postman Collection, opcional).

2. Frontend:
- Layout funcional e intuitivo.
- Consumo adequado da API.
- Responsividade e design (básico).

3. Extras (não obrigatórios):
- Testes unitários (JUnit no backend, Jest/React Testing Library no frontend).
- Deploy local com Docker Compose (opcional).

Entrega:

- Formato: Repositório no GitHub ou GitLab, com instruções para colocar a aplicação a funcionar no ficheiro README.md.