

Teste técnico - Desenvolvedor Fullstack

Observação: Mesmo que você não consiga finalizar completamente o teste, tudo o que for desenvolvido será avaliado com atenção, considerando a qualidade do código, organização, clareza das decisões técnicas e lógica de implementação. Por isso, envie o que tiver conseguido construir, mesmo que incompleto.

Cenário

Você está atuando como desenvolvedor na equipe de uma consultoria especializada em redes de computadores, responsável por oferecer suporte técnico a diversos provedores de internet (ISPs). Esses provedores frequentemente entram em contato com a consultoria para relatar problemas, solicitar diagnósticos de lentidão, agendar manutenções, configurar equipamentos como roteadores e BGPs, entre outras atividades técnicas. Atualmente, essas solicitações são registradas de maneira informal, como por meio de planilhas e e-mails, o que dificulta o acompanhamento do histórico, o controle de status e a documentação técnica.

Diante disso, a consultoria decidiu desenvolver um sistema interno simples que centralize a gestão das demandas dos provedores. O objetivo é que qualquer membro da equipe consiga cadastrar um provedor, registrar uma nova demanda, documentar as ações realizadas, atualizar o status e acompanhar todo o histórico de forma rápida e clara.

Exemplo prático

Um provedor chamado "BRNX Fibra" entra em contato solicitando uma análise de lentidão em sua rede de borda. Um atendente do monitoramento BRNX registra essa solicitação como uma nova demanda do tipo "Diagnóstico", com status inicial "Pendente". Um consultor analisa a situação, verifica tráfego elevado na interface `eth1`, aplica controle de banda e documenta essas ações no sistema. Após finalizada a intervenção, a demanda é marcada como "Concluída", ficando com o histórico completo disponível para consultas futuras. Ao final, o monitoramento repassa para o cliente o relatório da atividade.

Objetivo do sistema

O sistema a ser desenvolvido deverá permitir o cadastro dos provedores atendidos pela consultoria, o registro de novas demandas técnicas associadas a esses provedores e o acompanhamento do status e do histórico de ações realizadas em cada demanda. A aplicação deve conter tanto backend quanto frontend. Você tem liberdade para utilizar as tecnologias que considerar mais adequadas; no entanto, vale destacar que, em grande parte dos nossos projetos, utilizamos **Node.js e TypeScript no backend, React no frontend e PostgreSQL como banco de dados**, com o apoio do **Docker e Docker Compose** para orquestração e padronização do ambiente de desenvolvimento.

Escopo e requisitos funcionais

O sistema deve fornecer uma interface funcional que permita o cadastro de provedores, o registro de demandas técnicas vinculadas a esses provedores e o acompanhamento completo do status e do histórico de ações realizadas em cada demanda. Fica a seu critério a escolha das ferramentas e do estilo de implementação, contanto que as funcionalidades abaixo estejam contempladas.

Cada provedor deve conter informações básicas como nome fantasia, nome do responsável, e dados de contato. As demandas devem estar associadas a um provedor e conter, no mínimo, um título, uma descrição detalhada, o tipo de solicitação (como "Diagnóstico", "Manutenção", "Configuração", "Instalação" ou "Outro"), o status atual e a data de criação. O sistema também deve permitir o registro de ações realizadas em cada demanda, com campos como descrição da ação, nome do técnico responsável e data de execução.

A interface do sistema deve permitir visualizar a listagem de demandas em formato claro (como tabela ou cards), com opção de filtrar por status e por provedor. O usuário deve conseguir registrar novas demandas, visualizar os detalhes de uma demanda específica, consultar o histórico de ações técnicas e adicionar novas ações. Também deve ser possível atualizar o status da demanda de forma prática.

Tecnologias e estrutura esperada

Recomendamos que o backend seja desenvolvido em **Node.js com TypeScript**, utilizando o **Prisma ORM** para interação com um banco de dados **PostgreSQL**, e que o frontend seja implementado com **React e TypeScript**, com ou sem o uso de frameworks como o **Next.js**. Também sugerimos uma arquitetura organizada, com separação clara de responsabilidades entre **controllers**, **services** e **repositórios**, ou conforme o padrão arquitetural com o qual você tiver mais familiaridade.

Embora você tenha liberdade para definir a estrutura do projeto, recomendamos fortemente o uso do Docker. O **Docker** e do **Docker Compose** permitirá isolar os serviços (frontend, backend e banco de dados) e garantir que o sistema possa ser executado com poucos comandos. É importante incluir um arquivo **.env.example** com as variáveis de ambiente necessárias e instruções claras no **README.md** sobre como executar o sistema.

Banco de dados

O sistema deve contemplar três conceitos principais: **provedores**, **demandas** e **ações técnicas**. Fica a seu critério a forma como essas entidades serão modeladas, bem como os nomes e estruturas adotados. De forma geral, espera-se que cada provedor possua informações como nome fantasia, responsável, e meios de contato. As demandas devem estar vinculadas a um provedor e conter um título, descrição, tipo (como "Diagnóstico", "Manutenção", etc.), status atual e data de criação. Já as ações técnicas devem registrar intervenções realizadas em uma demanda, incluindo uma descrição, o nome do técnico responsável e a data da execução.

Você pode modelar o banco da forma que preferir. O mais importante é que as relações entre as entidades estejam bem definidas e façam sentido dentro da proposta do sistema.


README e entrega

O repositório deverá conter um README claro e bem estruturado, com instruções para rodar o projeto, explicação da estrutura do sistema e tecnologias utilizadas. O projeto deve ser hospedado em um repositório público no GitHub, com commits organizados e uso adequado de mensagens.

Funcionalidade adicionais

Funcionalidades adicionais que não foram explicitamente solicitadas, mas que contribuam para a usabilidade, organização, rastreabilidade ou eficiência do sistema, serão muito bem-vindas e valorizadas. Isso pode incluir melhorias na interface, recursos de autenticação, exportação de dados, logs de atividade, notificações, testes automatizados, entre outros. Sinta-se à vontade para propor e implementar qualquer funcionalidade extra que demonstre sua capacidade de ir além do básico e pensar em soluções práticas para o contexto proposto.

 **Que os seus commits sejam limpos, suas queries performáticas, e seus containers sempre subam na primeira tentativa!**

 Desejamos a você **boa sorte no teste**, jovem padawan do código! Que a força esteja com você — e seu linter também. 