Support Engineer

Conhecendo você melhor - Nome: Julia Barcelos

→ Chegou à você um problema de outro time, porém que você sabe resolver, o que você faz?

Confirmo com a liderança dos times se foi direcionado a pessoa certa. Se sim, faço.

→ Chegou à você três demandas ao mesmo tempo, como você prioriza esses atendimentos? Por quê faria desta forma ?
As demandas são as seguintes:

- Uma pessoa da área de Customer Success pedindo ajuda para entender um problema que um cliente está sofrendo; >> *IMPORTANTE, MAS NÃO URGENTE*
- Uma pessoa de suporte N1 reportando que a funcionalidade de Automação está fora do ar; >> *IMPORTANTE E URGENTE*
- Um gerente pedindo para você coletar dados sobre os atendimentos dos últimos 30 dias; >> *IMPORTANTE, MAS NÃO URGENTE*

- Escolho atender a pessoa do suporte N1, para juntos tentarmos resolver e subir a operação.

- Comunico o meu gerente que estou com uma demanda de urgência e tenho um caso customer e peço para que ele me sinalize a urgência dessa solicitação.

- Peço a pessoa do Customer Success montar um dossiê dos ocorridos pra eu conseguir avaliar com calma assim que eu terminar de auxiliar o suporte da N1.

  Acredito que dessa forma eu consiga dar um posicionamento da minha situação para todos e atender a única demanda declarada como urgente até o momento.

→ Você já teve alguma experiência com atendimento suporte/público que te marcou? Conta um pouco para nós como foi.

Um dos meus primeiros trabalhos foi sendo recepcionista de academia.
Depois fui recepcionista de escola de idiomas também.
Em ambas as experiências eu precisei contornar muitas situações.
Não tenho nenhuma de tão marco. Quando o assunto é financeiro, muitas vezes é difícil de resolver, no entanto, sempre consegui resolver e manter o cliente.

→ Fale um pouco sobre como o Developer Tools dos Navegadores (i.e. , firefox, chrome) podem ajudar na investigação de bugs em uma aplicação web.

Falando em CSS:
  Já ajuda demais você conseguir visualizar vários tamanhos de telas, simular que está acessando de outros dispositivos e manipulando a tela, para quebras de layout isso é uma mão na roda.
  Além disso, você pode simular suas edições e visualizar em tempo real, para descobrir se a sua ideia solucionará o problema, antes de sair editando o código fonte e se perder em meio a edições. O que pode ser bem útil no desenvolvimento de animações.
  O auto complete é algo que também ajuda muito a verificar se é algum erro de escrita do código.
Recentemente eu descobri essas maravilhas que são o Styles :how e Styles .cls onde você consegue verificar os hover, focus, active, visited e busca por temas definidos em classes. Para quem utiliza metodologia BEM, é super legal.

Falando em JS:
  Verificar retornos, fazer consultas de logs, ou consulta de trace quando você ainda não conhece o código. E melhor que console.log, só o debugger ou a combinação dos dois. E é só escrever ele no seu código, onde você suspeita da quebra e ir rodando o código step by step, ele vai te dando o valor real das constantes, ou retorno das funções. É também possível debugar por modificações de DOM e eventListener.

Falando da página como um todo, a Analyzing load performance pode te dar uma avaliação completa dos problemas encontrados na página e ai você consegue entender eles vasculhando a documentação. Não utilizei ainda, porque só trabalhei com páginas pequenas e explorando buscas por mim mesma.

→ Você prefere utilizar bancos de dados relacionais ou não relacionais em uma aplicação ?

Não preciso preferir, o que for melhor para aplicação deve ser utilizado.
Se forem dados mais sensíveis e com relacionamentos entre tabelas, o relacional parece interessante. Agora se os focos forem escalabilidade, agilidade na execução e no desenvolvimento o não relacional parece uma escolha melhor. Cada caso é um caso, sem bala de prata.

→ Como você explicaria para uma pessoa não técnica o funcionamento de uma requisição POST e GET?

A tradução das palavras pode ajudar a entender.
Post : é uma palavra do vocabulário inglês que se refere a *início, fixar e colar*.
Get : é uma palavra do vocabulário inglês que se refere a *pegar, levar e conseguir*.

Quando você quer incluir um dado no seu banco de dados, 'fixar', 'colar' essa informação na tabela e é a primeira vez (início) que você a colocará lá, utilize o método POST.

Quando você quer consumir um dado no seu banco de dados, 'conseguir', 'pegar' essa informação da tabela e levar até o service para manipular, utilize o método GET.

→ Um cliente entrou em contato reportando que modificou um dado de um Lead no RD Station e esse dado ainda não consta nos relatórios. Você verificou que a fila de processamento de alteração de dados em Leads está com uma latência alta. Como você explicaria isso para um cliente ?

Transparência e cuidado, explicaria a situação e já sinalizava o gerente do setor para montarmos um plano de ação para desafogar essa fila.

→ Se você fosse desenvolver um software do zero, quais os principais passos que você imagina que seriam necessários até chegar em um MVP para ser utilizado pelo cliente final ?

Depende, não ficou claro qual o desejo do cliente com o software em si para eu conseguir definir o que é o mínimo produto viável de entrega.

De forma geral o ideal é dividir em funcionalidades menores e programar entregas regulares com updates. Só que é preciso entender a necessidade inicial do cliente e seguir complementando as necessidades.
O que ele precisa primeiro é de um BD de fácil acesso?
Ou ele precisa de uma plataforma para consumir um BD que já existe?
É um software de uso interno, ou precisa de uma boa experiência de UI/UX para o desenvolvimento desde o início? Quem dita as necessidades é o cliente, cabe a nós definir a viabilidade. É importante saber que mais vale um feito do que o perfeito. Melhorias podem ocorrer após a primeira entrega, o produto não está finalizado.