# Questionário Técnico para Desenvolvedor – Jonatas Viana

## 1. Habilidades Obrigatórias

### Backend

• Você tem experiência com .NET 6 ou superior? O que você já desenvolveu com essa tecnologia?  
Sim, já trabalhei com .NET em um projeto com microsserviços, entretanto, a versão utilizada era inferior a versão 6, era .NET framework 4.5 com Dapper. Neste projeto, contribuí com o desenvolvimento de uma api para um banco.

### Frontend

• Você já trabalhou com Angular 16 ou superior ou React? O que você já desenvolveu com essas tecnologias?  
Já trabalhei com AngularJS, Angular 7, Angular 18 e 19, e neste momento, trabalho com Angular 14. Neste projeto trabalho no desenvolvimento de um sistema bancário.  
Também já trabalhei com React e React Native. No último projeto que atuei, trabalhei para um cliente do setor varejista do qual atuava no desenvolvimento da parte financeira do projeto. Utilizávamos React JS e Next.js para a parte WEB e React Native para o mobile.

### Banco de Dados

• Você tem experiência com bancos de dados relacionais (PostgreSQL, MSSQL) e/ou NoSQL (Redis, MongoDB)? O que você já fez com essas tecnologias?  
Dos bancos citados, já trabalhei com PostgreSQL, SQL Server, e Mongo.  
Com PostgreSQL trabalhei a bastante tempo, em um sistema de credenciamento de eventos feito com PHP.  
SQL Server trabalhei nesse projeto que citei antes, com .NET 4.5.  
MongoDB trabalhei no projeto que citei de React Native. O banco que o app utilizava era MongoDB (não me recordo exatamente, mas acredito que com Atlas), e Compass para interface.

### Arquitetura e Design de Aplicações

• Você compreende o conceito de aplicações orientadas a eventos? O que você já construiu? Qual foi o maior desafio que enfrentou?  
Sim, são aplicações que reagem a mudanças através de eventos disparados, de forma assíncrona e desacoplada de execuções.

• Você tem experiência com Azure Service Bus? O que você já desenvolveu?  
Sim e não. No teste que fiz e os enviei para avaliação, fiz uma pequena implementação de utilização do Azure Service Bus na API. Algo simples mesmo, um evento que dispara o envio de e-mail ao criar um usuário. Só quis demonstrar que, sei o que é esse serviço, para que serve, e como implementar. Porém, não possuo vasta experiência profissional com isso.

• Você já utilizou Entity Framework ou Dapper? Quando e em qual projeto? O que mais gosta nesse framework?  
Sim, já utilizei ambos. O projeto que citei com .NET 4.5 não utilizava Entity Framework, somente Dapper. Neste mesmo projeto, foi iniciado uma migração para o, na época, .NET Core, da qual eu atuei muito pouco. Neste projeto, se bem me lembro, foi implementado o EF, mas como disse, não mexi muito com isso. Na aplicação de teste que fiz e os enviei para avaliação, fiz com .NET 8 utilizando EF Core 8, então já um bom exemplo de implementação minha. Quanto a gosto, acredito que gosto de ambos, e acho que cada um tem suas vantagens e desvantagens. Por exemplo, no projeto que citei, tenho plena certeza de que optaram por utilizar Dapper pois as querys eram extremamente robustas, com muitas procedures envolvidas, o que tornaria o uso do EF um pouco mais complicado. Porém, o EF torna, de certa forma, a vida mais simples em alguns aspectos. Certamente para projetos novos, ou com escopo menor (como esse do meu teste), vale muito a pena utilizar. Eu, pessoalmente, gosto muito de utilizar Node.js com Mongo. E como podem ver neste meu [projeto do github (shopping-list-api)](https://github.com/jonatasmviana/shopping-list-api-open), utilizo de uma abordagem “semelhante”, onde não faço uso de querys, e sim do framework (insertOne, por exemplo).

• Qual o nível de maturidade do REST? Você pode explicar os diferentes níveis?  
A maturidade REST vai do nível 0 ao nível 3. Porém, profissionalmente falando, nunca vi uma API implementada no nível 3, sempre param no nível 2, pois já oferece um bom equilíbrio entre simplicidade e aderência ao REST.  
Nível 0 – Uma API com URL única, que usar HTTP somente como transporte, e tudo é enviado via POST.  
Nível 1 – Segue fazendo tudo via POST, igual no nível 0, porém adicionando recursos, como URLs/endpoints distintos.  
Nível 2 – Utiliza os verbos HTTP corretamente, com POST, PUT, GET, DELETE, para representar as ações. Também implementa apropriadamente os status codes (200, 404, 500, etc).  
Nível 3 – Sinceramente, tive que pesquisar a respeito, pois sabia da existência deste nível, mas não sabia explicar. Mas basicamente são APIs que fornecem links dentro de suas respostas, para indicar próximas ações, permitindo que os clientes naveguem pelos recursos sem depender de conhecimento prévio da estrutura da API.

## 2. Testes e Qualidade de Código

• Você sabe criar testes unitários? Quais ferramentas você utiliza? (O candidato deve mencionar xUnit ou nUnit)  
Sim, trabalho diariamente com criação de testes unitários, porém, no front-end. Utilizo de frameworks como Jest, Jasmine e Karma. No back-end, infelizmente, não tenho experiencias com testes unitários.

• Você tem experiência com revisão de código?  
Sim, trabalho diariamente fazendo revisão de código de colegas.

- O que você analisa ao revisar um código?  
Procuro analisar sempre se o código é fácil de entender, se possuí nomes de variáveis, funções/métodos descritivos. Se o código está bem estruturado, evitando repetições e complexidade desnecessária. Tento ver se o código é eficiente ou se existem gargalos desnecessários.

- Como faz sugestões a um colega?  
Gosto de dar sugestões por meio da ferramenta de code review mesmo, ou falando diretamente com o colega, principalmente se não entendi algo, aí falando direto com o colega, ele pode explicar melhor.

- Como lida com feedbacks sobre o seu código?  
Lido de forma bem tranquila. Gosto de encarar todo feedback nesse sentido como algo construtivo, afinal, ninguém faz (ou ao menos não deveria fazer) code review com más intenções né. Então, se alguém comentou algo, posso concordar e fazer a alteração, ou discutir, conversar a respeito, para que se possa ter o melhor resultado final possível.

## 3. Arquitetura de Software e Metodologias

• Você tem experiência com microserviços? Como foi a aplicação?  
Sim, no projeto que citei com .NET 4.5 era utilizado a arquitetura de microserviços. A aplicação em si não saberia dizer, pois quando entrei no projeto, a estrutura já estava toda criada. Mas eu trabalhava diretamente com a aplicação, as vezes tendo que rodar vários MSs ao mesmo tempo, aplicando as dependências locais em cada um deles.

• Você conhece Gitflow? Pode explicar?  
É um modelo para o Git que organiza o desenvolvimento em branches, aplicando um fluxo do qual ajuda a organizar o trabalho em equipe e garantir um ciclo de desenvolvimento mais controlado. Normalmente é definido em branchs main, develop, feature, release e hotfix.

• Você tem conhecimento sobre DDD (Domain-Driven Design)? Pode explicar?  
É uma abordagem de desenvolvimento que tem foco no domínio, nas regras de negócio. A ideia é modelar o software de forma próxima à realidade do negócio. No projeto em node que citei acima, eu fiz algo com elementos de DDD, mas ainda não ficou totalmente alinhado com o design correto.

• Você já trabalhou com metodologias ágeis? Quais? Como funcionavam no seu time?  
Sim, a muito tempo já trabalho com metodologias ágeis. Desde que comecei a atuar com essas metodologias, trabalhei basicamente com uma mescla de Scrum com Kanban. Scrum, pois sempre trabalhamos com sprints, tendo ciclos curtos de planejamento/desenvolvimento/teste/entrega, tendo os papéis bem definidos, como Scrum master, PO, e time de desenvolvimento. E Kanban, pois sempre tivemos a visualização do fluxo de trabalho por meio de um quadro, seja utilizando um quadro de fato com post its, ou utilizando o board de aplicações, como Azure, Trello, etc.

## 4. DevOps e Integração Contínua

• Você já utilizou Azure DevOps? Sabe o que é?  
Sim, trabalho com Azure a algum tempo já. Azure DevOps é um conjunto de ferramentas da Microsoft para gerenciar todo o ciclo do desenvolvimento. Com ele você tem controle de versionamento, integração contínua, gerenciamento de projetos ágeis, testes automatizados.

• Você sabe o que é CI/CD (Integração Contínua e Entrega Contínua)? Pode explicar?  
O CI/CD são práticas de DevOps onde se aplicam processos e automações para agilizar o desenvolvimento e o deploy de software. A integração contínua (CI) automatiza a compilação e testes do código sempre que há alterações. E a entrega contínua (CD) prepara e implanta automaticamente o código nos ambientes.

## 5. Princípios de Desenvolvimento

• Você sabe o que é injeção de dependência? Para que serve?  
Injeção de Dependência é um padrão de design que permite que um objeto receba suas dependências de fora, em vez de criá-las. Essas dependências podem ser injetadas por construtors, setters ou interfaces. Pensando também em front-end, uma forma de aplicar DI é com componentes, pois um componente normalmente tem a intenção de ser reutilizável, portanto, injetando dependências nele por meio de parâmetros, o torna também um bom exemplo de injeção de dependência. Inclusive no Angular, em versões mais antigas, se usa construtores para essas injeções semelhante a linguagens como C#. E em versões mais novas, se pode usar diretamente o inject.

• Você conhece o conceito "Open/Closed"? Como o aplica no seu dia a dia?  
Sim, conheço o conceito Open/Closed, o O do SOLID. No meu dia a dia, aplico esse princípio escrevendo código que seja aberto para extensão, mas fechado para modificação. Por exemplo, ao lidar com notificações em Angular, utilizo injeção de dependência para criar implementações sem alterar o código existente. Se precisar adicionar um novo tipo de notificação, basta criar um novo serviço que implemente a mesma interface, sem modificar as classes já existentes.

• Você conhece o conceito "Single Responsibility"? Como o aplica no seu dia a dia?  
Sim, conheço o conceito e aplico esse princípio garantindo que cada classe ou função seja responsável por uma única tarefa. Por exemplo, em Angular, separo a lógica de negócios dos componentes criando services para lidar com chamadas HTTP ou manipulação de dados. Assim, os componentes ficam focados apenas na interface do usuário.

• Como é o seu dia a dia como desenvolvedor? Como você gerencia seu tempo?  
Meu dia a dia como desenvolvedor envolve planejamento, desenvolvimento e colaboração. Sou uma pessoa bem metódica, muito organizado, então gosto de seguir rotinas. Costumo iniciar e finalizar meu dia vendo meus e-mails e compromissos, para ver quais reuniões e agendamentos tenho para o dia, e para ver se não possuo nenhum apontamento logo cedo no dia seguinte. Também leio e respondo tudo que precisa, caso tenha algo. Após isso, verifico se não existem Pull Requests abertos, para review. E na sequência dou início ao meu trabalho de fato no desenvolvimento.  
Geralmente, começo organizando minhas tarefas com base nas prioridades definidas no time. Quando inicio uma tarefa, costumo ler todo enunciado, e tentar entender o máximo que deve ser feito. Se eu tiver dúvidas, anotarei para tirar essas dúvidas na daily. Claro que, se for algo muito pontual, ou de caráter impeditivo, tento resolver na hora. Tudo isso para ter uma melhor otimização de tempo dentro do meu dia de trabalho.