

Prefácio

por Jim Webber*

O ARQUITETO

Quando penso sobre o termo “arquiteto”, é sempre com sentimentos confusos, como suspeito que seja o caso para muitos de nós que trabalham com desenvolvimento de software atualmente. Durante minha carreira, ouvi diversas opiniões diferentes para esse termo, mas, para mim, o que é principal para ser um bom arquiteto posso buscar dos meus primeiros dias como desenvolvedor profissional (ou tentando ser um profissional).

Meu primeiro emprego parecia ser intimidador – após diversos anos de pesquisa acadêmica dentro de um departamento de ciência de computação em uma universidade, entrei em uma empresa que havia comprado recentemente uma *startup* de *middleware* Java gerenciada por um grupo de amigos. Esses amigos trabalhavam no ramo de *middleware* por décadas e “hackeavam” em Java até mesmo antes de ele ter esse nome. Sair da zona de conforto da computação de alto desempenho, com tecnologias as quais eu estava acostumado, e pular para um domínio novo de processamento de transações distribuídas com foco em CORBA e J2EE era realmente intimidador. Apesar de ter um senso razoável de teoria de sistemas distribuídos, eu jamais havia escrito CORBA ou J2EE como usuário, muito menos como desenvolvedor de *middleware*. Foram tempos complicados.

No laboratório onde comecei a trabalhar, meus colegas ostentavam nomes de cargos baseados em suas capacidades (e, suspeito, longevidade!). Possuíamos engenheiros, engenheiros seniores e arquitetos, todos os quais trabalhavam arduamente em desenvolver um excelente *middleware* de transação, *workflow* e men-

* Jim Webber é formado em computação e possui doutorado pela University of Newcastle. Trabalhou por bastante tempo na ThoughtWorks, onde ficou conhecido pelo seu manifesto Guerrilla SOA. Hoje, é Chief Scientist na Neo Technology, empresa por trás do banco de dados orientado a grafos Neo4j.

sageria. Como um engenheiro sênior, eu era um pouco menos verde do que os apenas engenheiros, mas ambos recebíamos ordens dos arquitetos – um por time – que eram oráculos. Os arquitetos que lideravam nosso time conheciam o código de trás para a frente, mas também conheciam seus campos. Era sempre possível confiar que seriam capazes de explicar modelos de transação distribuída complexos e mostrar como diversos padrões e abstrações do código os suportavam. Eles também conheciam as armadilhas e conseguiram detectar cedo escolhas de design e implementação ruins, uma vez que eram capazes de ver à frente, ou pelo menos ver mais longe do que eu conseguia.

Não acho que disse algum dia para esses arquitetos – talvez por causa do meu orgulho juvenil – o quão importante eles foram em delinear meu pensamento e comportamento. Entretanto, sempre aspirei (e algumas vezes sucedi) ser como eles, conhecer a tecnologia profundamente, entender o contexto no qual ela está sendo empregada, guiar e ser guiado pelos colegas de meu time. Essa foi uma aspiração que levei para meu próximo emprego em uma empresa global muito conhecida e admirada no ramo de consultoria de TI.

Ao entrar em minha nova empresa, decidi que já havia visto bastante software e era bom o suficiente em escrever código, e então me declarei um arquiteto. Ao encontrar alguns de meus novos colegas pela primeira vez, fui calorosamente recebido e perguntado sobre qual seria meu papel. Respondi que eu era um arquiteto; algo que pensei que daria um sentido de competência no nível micro da pilha de tecnologias, e no nível macro ao redor da construção de sistemas. Fiquei atônito ao descobrir que fui repreendido por essa resposta e fiquei me perguntando o que acabara de acontecer, afinal, eu não havia entrado para trabalhar com essas pessoas devido a reputação que eles possuíam no desenvolvimento de software e Agile?

Não demorou muito para que eu entendesse o contexto. Meus colegas trabalhavam com consultoria há alguns anos e conheceram muitos “arquitetos experientes”. Mas esses não eram o mesmo tipo de pessoa que eu admirava em meu primeiro trabalho; eram arquitetos perigosamente ultrapassados, com pouca habilidade em entregar software, e um talento sem fim para causar confusão e criar políticas corporativas. E eu, aparentemente, tinha me apresentado como um desses. Não era a melhor maneira de causar uma boa impressão aos meus novos colegas.

Ainda assim, durante os 6 anos e nos 10 países em que trabalhei, me descrevi como arquiteto. Eu estava em uma missão de recuperar o valor desse título, e com o passar do tempo obtive algum sucesso. Começamos a pensar em um arquiteto não só como o “melhor programador” ou um “peso morto”, mas sim como um

papel que considera os detalhes de nível macro em um time que entrega software – como a integração com outros sistemas funcionará, como falhas e recuperações serão trabalhadas, como a vazão será medida e garantida.

Nos melhores projetos em que trabalhei, o papel do arquiteto era complementar ao dos *stakeholders*. Enquanto o pessoal de negócios prioriza a entrega de funcionalidades, arquitetos priorizam os requisitos não funcionais e ajudam a ordenar a entrega de requerimentos funcionais que minimizam e mantêm características não funcionais. Nos melhores projetos que trabalhei, eu não era o melhor programador da equipe, nem na maior parte das vezes tinha o maior conhecimento das linguagens, ferramentas e *frameworks* comparado com outros dedicados programadores. Após um tempo engolindo o orgulho, comecei a ver as vantagens inerentes disso.

Como arquiteto, ainda sou alguém que também entende bastante de código, em um nível no qual consigo escrever código com outros programadores (possivelmente melhores do que eu) e ler seu código e testes. Acredito que isso seja fundamental para a arquitetura. Sem a habilidade de se envolver com o código, arquitetos não têm como determinar como sua arquitetura foi adotada por uma solução, nem como eles têm uma maneira direta de pegar *feedback* de sua arquitetura para melhorá-la. Sem habilidades de software, um arquiteto fica refém dos times de desenvolvimento, o que é um *overhead*, não um benefício.

Sentado aqui em minha mesa, em Londres, escrevendo esse prefácio, é claro que a noção do conceito de arquiteto continua cinzenta. Até mesmo hoje, arquitetos estão frequentemente no topo da torre de marfim, descrevendo visões grandiosas e impraticáveis através de slides e diagramas UML que não ajudam para tropas nas trincheiras obedecerem como escravos (como se eles fossem!). Mas não precisa ser assim. Aqueles de nós que se preocupam com a melhoria contínua do ramo e se preocupam em escrever código cada vez melhor com os efeitos colaterais positivos que isso pode resultar em nossa sociedade e comunidade têm o dever de tomar o termo “arquiteto” e reapropriar-se dele.

Este livro é um passo nessa direção. Ele trata arquitetos como profissionais do desenvolvimento de software e é fundamental que eles entendam as ferramentas comuns do ramo de desenvolvimento de software. Este livro provê *insights* nas ferramentas comuns, padrões de desenvolvimento e práticas de design que arquitetos contemporâneos trabalhando em aplicações corporativas possuem. Ele trata de uma gama de disciplinas de desenvolvimento desde código robusto, design de aplicação, dados e integração, que são os principais elementos da maior parte dos sistemas corporativos. Ele não apenas foca em importantes detalhes da linguagem

e dos principais *frameworks*, mas também no nível macro, dando uma percepção mais completa – percepção esta que um arquiteto praticante precisa quando ajuda a guiar um time para o sucesso.

À medida que desenvolvedores procuram se tornar arquitetos, o livro mostra que o foco estreito na última linguagem ou *framework* da moda é necessário, mas não suficiente. À medida que arquitetos buscam reabilitação, este livro deixa óbvio que não há sucesso sem reaprender as (habilmente demonstradas) ferramentas de construção de software. Enquanto nenhum livro pode torná-lo um arquiteto da noite para o dia, este provê ao menos um modelo do tipo de arquiteto que eu tanto admirava, e espero que sua leitura o inspire a se tornar esse tipo de arquiteto também.