Vibe Coding: menos código, melhores prompts?

digitalinside.sapo.pt/vibe-coding-menos-codigo-melhores-prompts

Na realidade, vibe coding descreve uma mudança na forma como o software está a ser criado. Em vez de o programador escrever código linha a linha, passa a descrever intenções, contexto, objetivos e comportamentos desejados a sistemas de Inteligência Artificial que, por sua vez, geram o código, sugerem arquiteturas e até podem ajudar a resolver problemas. O foco deixa de estar na sintaxe e passa a estar na clareza da intenção. No fundo a premissa, em tom de exagero, é de se deixar de programar uma máquina para se conversar com uma máquina que programa por nós.

Isto muda completamente o papel de quem desenvolve software. Já não se trata apenas de dominar linguagens ou frameworks, mas de saber explicar bem o que se quer construir, decompor problemas de forma inteligente e guiar a IA para resultados úteis. E é aqui que entra uma questão inevitável: sim, vibe coding implica, quase por definição, um novo tipo de prompt engineering. Não no sentido mais académico do termo, mas no sentido prático e diário de saber formular pedidos que fazem sentido para modelos de IA cada vez mais capazes, mas ainda dependentes de contexto claro e bem estruturado.

Os benefícios desta abordagem são evidentes, a começar pela velocidade com que ideias passam a protótipos, e a própria redução de tarefas mais repetitivas. Ao mesmo tempo, democratiza o acesso ao desenvolvimento de software, permitindo que pessoas sem formação técnica profunda consigam construir aplicações funcionais. Profissionais de diferentes áreas diferentes conseguem agora, por exemplo, criar aplicações mais adaptadas às suas necessidades reais de trabalho em vez de utilizar algo mais genérico.

Mas esta facilidade também esconde um lado menos confortável. Quando o código é criado por sistemas que não se compreendem totalmente, cresce o risco de construir sistemas frágeis, difíceis de manter ou simplesmente errados de forma silenciosa. E há ainda um efeito mais subtil: a perda gradual de compreensão profunda. Se tudo funciona porque a IA “resolve”, deixa de haver incentivo imediato para entender o que está realmente a acontecer por baixo do capô. Isso pode criar uma camada de utilizadores muito produtivos, mas menos conscientes da complexidade dos sistemas que estão a utilizar.

É assim uma ideia profundamente errada dizer que isto substitui o programador tradicional, sobretudo, quando falamos em termos de segurança, escalabilidade e arquitetura de sistemas. O que está a acontecer é mais uma mudança de função do que uma substituição. O trabalho desloca-se do ato de escrever código para o ato de orientar sistemas, validar resultados e tomar decisões arquiteturais. E quanto mais código for criado automaticamente, mais importante se torna a capacidade de pensar de forma crítica sobre esse código. Um modelo de IA consegue criar componentes e até aplicações

inteiras, contudo, continua sem compreender totalmente o contexto do negócio, as decisões e especificidades da arquitetura, os compromissos técnicos ou as consequências reais de um erro em produção. Só um programador consegue avaliar criticamente se aquele código faz sentido, se é seguro, escalável e sustentável a longo prazo. A IA pode acelerar desenvolvimento, reduzir tarefas repetitivas e aumentar produtividade, não obstante, continuará dependente de supervisão humana.

Bruno Castro é Fundador & CEO da VisionWare. Especialista em Cibersegurança e Análise Forense.