Boas práticas de programação – filosofias de desenvolvimento

Aug 18, 2011 • William Bruno

Vou escrever um resumão de alguns conceitos muito importantes para o desenvolvimento de softwares, independente da linguagem que você estiver ‘falando’.

Conceitos antigos, mas que muitas vezes não são tão difundidos quanto deveriam.

Para evitar uma confusão durante a leitura, e alguns comentários perdidos, informo que o nível desse artigo é mediano. Nem muito avançado, a ponto de que um Dev iniciante fique completamente perdido (talvez apenas não entenda alguns trechos, mas nada que atrapalhe a assimilação do todo), e nem tão básico, a ponto de ser desinteressante para quem já tem um pouco mais de experiência.

Um dos conceitos mais importantes, diz que o teus códigos devem ser de fácil manutenção. E é impossível que a tua aplicação seja ‘simples’ de refatorar, se ela tiver rotinas idênticas espalhadas “ao Deus dará”. Note que os conceitos que citarei, se aplicam a um mesmo projeto. Durante o desenvolvimento de uma aplicação, leve em conta os seguintes pontos:

SOC = Separation Of Concerns

Separe os interesses. Divida rotinas em trechos tão pequenos quanto possível, e tanto quanto ainda fizer sentido. Quando temos um grande problema para resolver, devemos começar separando esse problemão, em probleminhas menores, e resolver cada probleminha de uma vez. Isolando em pequenas partes focadas, você privilegiará até o reaproveitamento do seu código.

DRY = Dont Repeat Yourself

Não repita a si mesmo. Se você já programou um trecho de código que resolve uma dada situação, e logo mais adiante, se depara **com a mesma situação** novamente, você não deve desenvolver a mesma rotina novamente, ou muito menos replicar o script. Sem essa de Ctrl+C / Ctrl+V (ou command no caso de usar MAC). Também pode ser conhecido por:

DIE = Duplication is Evil

E é o que realmente é. Duplicação é ruim. Ruim para o desenvolvedor, para o desenvolvimento, para futuras correções.. Não duplique códigos.

Existem até Design Patterns específicos, que tratam desse tipo de questão.

Desacoplamento é o objetivo. Devemos saber que é impossível termos componentes completamente independentes, porém o Strategy por exemplo, pode nos ajudar nisso.

Trabalho com sistemas web. Programo hoje em dia principalmente em php. Não estou falando que *'é ruim'* eu instalar o WordPress 2~3, várias vezes para vários clientes, duplicando o core do WP em vários dominios. Digo para você não se repetir num mesmo site, num mesmo projeto.

Se eu já implementei uma rotina de conexão ao banco de dados, não vou duplicar essa rotina. Não vou ter mais que um único arquivo com os dados de configuração. Não vou jogar por vários cantos da aplicação a lógica do cálculo de descontos do Ecommerce. Tenho que analisar bem, e centralizar, para não me repetir, ou "reinventar a roda" que eu mesmo já inventei.

KISS = Keep It Simple, Stupid

Mantenha simples. Descarte toda a complexidade que não for realmente necessária.

Lógico que nem tudo é simples. Existem coisas que realmente não são. Porém a idéia do KISS, não é deixar fácil, mas sim evitar complicar. Evitar aumentar o custo, entregando exatamente o que foi requisitado. E apenas isso.

Você e sua aplicação só ganharão com isso.

KISS é fazer menos, e não fazer da forma mais fácil.

Mantenha o foco. Desse conceito, derivam os dois seguintes:

YAGNI = You Aren't Going Need It

Bastante parecido com o conceito de cima. *"Você não vai precisar disso"*, nos diz para não complicar, ou ficar inventando o que não será usado.

É uma tentação comum querermos desenvolver o sistema mais incrível do mundo, com as melhores funcionalidades.

Seja racional. Preveja a expansão e escalabilidade, porém não fique dando voltas atrás do próprio rabo, sem sair do lugar, ou entregar o que foi pedido, por estar implementando algo que não era nem requisito e nem necessário naquele momento.

Worse is better, também nos diz exatamente isso. A qualidade não provém somente de mais funcionalidades. Ao pé da letra *Quanto pior, melhor*. Devemos prezar pela praticidade e usabilidade, mesmo que sejamos obrigados a impor limites. Algumas expansões são óbvias, e não vamos deixar de fazê-las, porém analise se neste momento, é necessária ou não.

Além desses pontos, nunca devemos nos esquecer de identar, comentar, documentar..

Um não concorre com o outro. Use todos juntos. Como complementos. Aplicar um desses conceitos, te leva ao seguinte.

Ainda não tenho conhecimento suficiente para ensinar. Continuo estudando, e espero ter contribuido com alguma coisa (assim como contribui com o meu próprio aprendizado, ao me propor escrever esse texto), já que você teve paciência de chegar até aqui.

Obrigado por ter lido.

Lembra de mais alguma ‘máxima’? Me conte nos comentários.

• Responder • Partilhar ›

Thiago Cruz • há 6 anos

É bem isso ai mesmo.
Quanto mais você tenta complicar algo simples, mais você ta usando um tempo que você não tem, em algo que você não precisa.

Sempre mantenha seu código simples, fácil e pratico, não de voltas... e sempre que for utilizar algum código novalmente:
1 - tente sempre re escrever ele, normalmente você o vai fazer melhor
2 - tente não copiar todo ele caso seja necessário copiar... pois se existia algum problema antes esse problema pode re aparecer
3 - Nunca em tempo algum copie código (framework) feito por outros sem saber como usar a ferramenta... isso pode lhe fazer levar mais tempo fazendo algo que você poderia fazer em 5 minutos só porque você está tentando usar código alheio.

E é isso ai ;) mantenha sua vida simples e seja feliz.

Mto bom artigo ;)

Abraços

• Responder • Partilhar ›

Bem Vindo ao meu blog. Me chamo William Bruno (aka wbruno) e trabalho como Desenvolvedor NodeJS nos horários em que não estou estudando Matemática ou escrevendo posts.