

Código Limpo

Robert C. Martin

Capítulo 1: Código Limpo

Daniel Pereira Volpato

07/06/2022



A low-angle, upward-looking photograph of two modern skyscrapers with glass facades. The buildings are set against a pale, overcast sky. The perspective creates a sense of height and architectural scale. The glass reflects the sky and surrounding environment.

O Autor

Robert C. Martin (Uncle Bob)

3778



Profissional de softwares desde 1970 e consultor internacional de software desde 1990. Ele é o fundador e o presidente da Mentor Object Inc., uma equipe de consultores experientes que orientam seus clientes no mundo todo. Foi um dos fundadores do Manifesto Ágil.

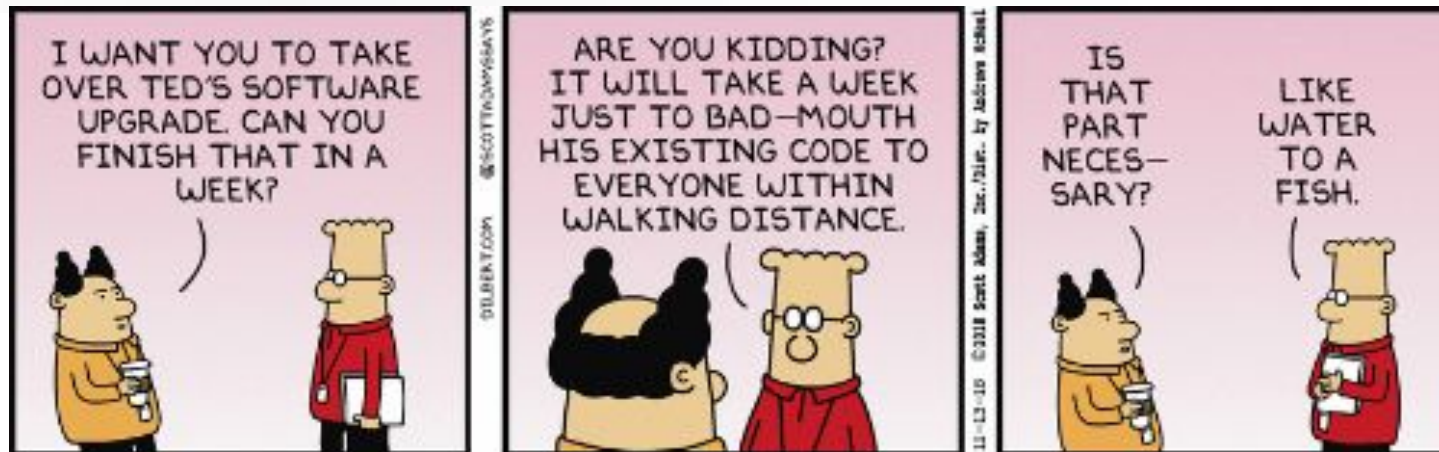
A low-angle, upward-looking photograph of two modern skyscrapers with glass facades. The buildings are angled towards the top right of the frame, creating a sense of height and scale. The sky is a pale, overcast grey. The word 'Introdução' is superimposed in the center in a large, bold, black font.

Introdução

O primeiro capítulo conduz o leitor a
**perceber o impacto de um código ruim para uma
organização**
e
como se portar diante da pressão por entregas rápidas
vs. um código limpo.

A low-angle, upward-looking photograph of two modern skyscrapers with glass facades. The buildings are slightly tilted, creating a sense of height and architectural scale. The sky is overcast and grey. The text 'Código Ruim e Confuso' is superimposed in the center in a large, bold, black font.

Código Ruim e Confuso



“um bom código importa, pois tivemos de lidar com a falta dele por muito tempo”

O Custo de um Código Confuso

```
//  
// Dear maintainer:  
//  
// Once you are done trying to 'optimize' this routine,  
// and have realized what a terrible mistake that was,  
// please increment the following counter as a warning  
// to the next guy:  
//  
// total_hours_wasted_here = 25  
//
```



```

1 function validate(str) {
2
3   if (str !== null && str !== undefined) {
4     if (str.length ≥ 11 || str.length ≤ 14){
5       str=str.replace('.', '').replace('-', '')
6         .replace('-', '').replace(" ", "");
7       if (!str.split("").every(c => c === str[0])) {
8         try{
9           let    d1, d2;
10          let    dg1, dg2, rest;
11          let    digito;
12          let    nDigResult;
13          d1 = d2 = 0;
14          dg1 = dg2 = rest = 0;
15
16          for (let nCount = 1; nCount < str.length -1; nCount++) {
17            // if (isNaN(parseInt(str.substring(nCount -1, nCount)))){
18              // return false;
19            // } else {
20
21              digito = parseInt(str.substring(nCount -1, nCount));
22              d1 = d1 + ( 11 - nCount ) * digito;
23              d2 = d2 + ( 12 - nCount ) * digito;
24            // }
25          };

```

```




26
27          rest = (d1 % 11);
28          dg1 = (rest < 2) ? dg1 = 0 : 11 - rest;
29          d2 += 2 * dg1;
30          rest = (d2 % 11);
31          if (rest < 2)
32            dg2 = 0;
33          else
34            dg2 = 11 - rest;
35          let nDigVerific = str.substring(str.length-2, str.length);
36          nDigResult = "" + dg1 + "" + dg2;
37          return nDigVerific === nDigResult;
38        }catch (e){
39          console.error("Erro !" + e);
40          return false;
41        }
42      } else return false
43    } else return false;
44  } else return false;
45 }
46
47 module.exports = {
48   validate
49 };

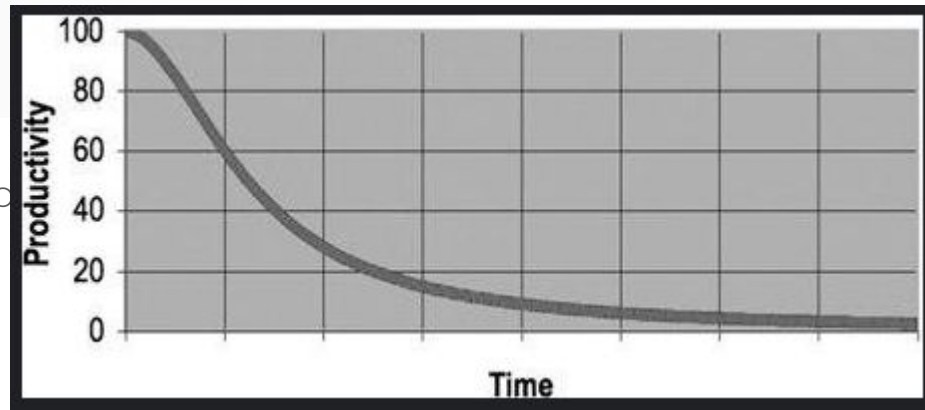
```

Fonte: Curso Clean Code e Clean
Architecture - Rodrigo Branas

Quem nunca...

3778

-  Equipe começa rápida, entregando muito
-  Depois de algum tempo começa a andar a passos de tartaruga
 - alteração em um ponto de código causa falha em 2 ou 3 lugares
 - nenhuma mudança é trivial!
 - adicionar ou modificar código exige passar por remendos e gambiarras
-  Tempo passa e a confusão aumenta ⇒ produtividade diminui cada vez mais



A “solução”

3778

- Solução da gestão:



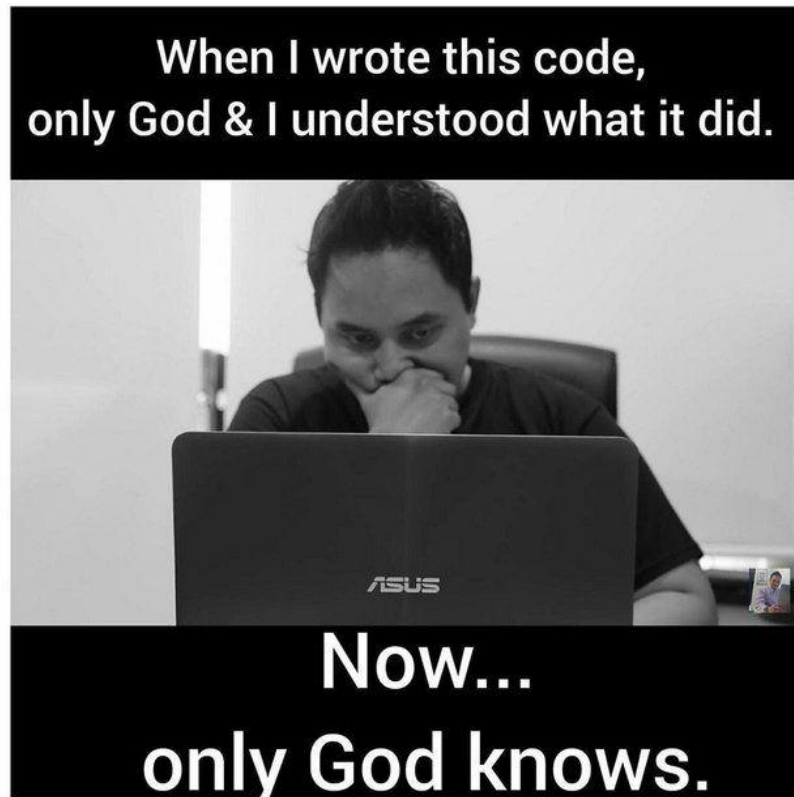
A low-angle, upward-looking photograph of two modern skyscrapers with glass facades. The buildings are slightly tilted, creating a sense of height and architectural scale. The sky is a pale, overcast grey. The text 'Mudança de atitude' is superimposed in the center in a large, bold, black font.

Mudança de atitude

Como poderia essa zona ser culpa nossa?

3778

“Nossa **cumplicidade** no planejamento do projeto é tamanha que compartilhamos de uma **grande parcela da responsabilidade** em caso de falhas; especialmente se estas forem em relação a um código ruim.”



Gestores protegem os prazos e requisitos com paixão

“A sua [função como dev] é proteger o código com essa mesma paixão.”

“não é profissional que programadores cedam à vontade dos gerentes que não entendem os riscos de se gerar códigos confusos.”

Late equals never!



1. Sabemos que bagunças anteriores reduzem o desempenho.
2. Mas nos sentimos pressionados a cometer essas mesmas bagunças para cumprir os prazos.

Onde está o erro?

2. um código bagunçado não ajuda a cumprir o prazo, mas reduz a velocidade de trabalho e nos faz perder o prazo.

“A única maneira de isso não acontecer — a única maneira de ir mais rápido — é sempre manter o código limpo.”

A low-angle, upward-looking photograph of two modern skyscrapers with glass facades. The buildings are angled towards each other, creating a sense of height and scale. The sky is a pale, overcast grey. The text 'Código Limpo' is superimposed in the center of the image.

Código Limpo

**Para você, o que é um
código limpo?**

“Gosto do meu código **elegante e eficiente**. A **lógica deve ser direta** para dificultar o encobrimento de bugs, as **dependências devem ser mínimas** para facilitar a manutenção, o **tratamento de erro deve ser completo** de acordo com uma estratégia clara e o desempenho deve ser próximo do mais eficiente, de modo a não incitar as pessoas a tornarem o código confuso com otimizações sorrateiras. O **código limpo faz bem apenas uma coisa.**”

— Bjarne Stroustrup, criador do C++

“Um código limpo é **simples e direto**. Ele é tão bem legível quanto uma prosa bem escrita. Ele **jamais torna confuso o objetivo** do desenvolvedor. Pelo contrário, ele está repleto de abstrações claras e linhas de controle objetivas.”

— Grady Booch, autor de *Object Oriented Analysis and Design with Applications*

“Código limpo pode ser **lido e melhorado por um outro desenvolvedor**, que não seja seu criador. Ele tem testes de unidade e de aceitação e nomes significativos; ele oferece apenas uma maneira, e não várias, de se fazer uma coisa; possui **poucas dependências**, as quais são **explicitamente declaradas** e oferecem uma API mínima e clara. O código deve ser **inteligível** pois, dependendo da linguagem, nem toda informação necessária pode ser expressa no código em si.”

— Dave Thomas, fundador da OTI e pai do Eclipse

“Eu poderia listar todas as qualidades que vejo em um código limpo, mas há uma predominante que leva a todas as outras. **Um código limpo sempre parece que foi escrito por alguém que se importa.** Não há nada óbvio que se possa fazer para torná-lo melhor. **Tudo foi pensado** pelo autor do código, e se tentar pensar em algumas melhorias, você acabará voltando ao início, ou seja, **apreciando o código deixado para você** por alguém que se importa bastante com a criação [‘craft’].”

— Michael Feathers, autor de *Working Effectively with Legacy Code*

“Você sabe que está criando um código limpo quando **cada rotina que você lê se mostra como você esperava**. Você pode chamar de código belo quando ele também **faz parecer que a linguagem foi feita para o problema.**”

— Ward Cunningham, co-criador da XP, líder da Smalltalk e da Orientação a Objetos

Um exemplo

3778

```
1  const FACTOR_DIGIT_1 = 10;
2  const FACTOR_DIGIT_2 = 11;
3  const MAX_DIGITS_1 = 9;
4  const MAX_DIGITS_2 = 10;
5
6  function validate(cpf = "") {
7    cpf = extractDigits(cpf);
8    if (isInvalidLength(cpf)) return false;
9    if (isBlocked(cpf)) return false;
10   const digit1 = calculateDigit(cpf, FACTOR_DIGIT_1,
    MAX_DIGITS_1);
11   const digit2 = calculateDigit(cpf, FACTOR_DIGIT_2,
    MAX_DIGITS_2);
12   let calculatedCheckDigit = `${digit1}${digit2}`;
13   return getCheckDigit(cpf) === calculatedCheckDigit;
14 }
15
16 function extractDigits(cpf) {
17   return cpf.replace(/\D/g, "");
18 }
19
20 function isInvalidLength(cpf) {
21   return cpf.length !== 11;
22 }
23
```

```
24 function isBlocked(cpf) {
25   const [digit1] = cpf;
26   return cpf.split("").every(digit => digit === digit1);
27 }
28
29 function calculateDigit(cpf, factor, max) {
30   let total = 0;
31   for (const digit of toDigitArray(cpf).slice(0, max)) {
32     total += digit * factor--;
33   }
34   return (total%11 < 2) ? 0 : (11 - total%11);
35 }
36
37 function toDigitArray(cpf) {
38   return [...cpf].map(digit => parseInt(digit));
39 }
40
41 function getCheckDigit(cpf) {
42   return cpf.slice(9);
43 }
44
45 module.exports = {
46   validate
47 };
```

Fonte: Curso Clean Code e Clean
Architecture - Rodrigo Branas

A low-angle, upward-looking photograph of two modern skyscrapers with glass facades. The buildings are filled with windows, reflecting the sky. The perspective creates a sense of height and scale. The word 'Princípios' is overlaid in the center in a large, bold, black font.

Princípios

• 10:₁

- Proporção entre leitura e escrita de código
- Escrever código implica em ler e entender o contexto adjacente

**“se quiser que seu código seja de fácil escrita,
torne-o de
fácil leitura.”**

Janelas quebradas e código sujo

3778



Regra de escoteiro

3778



Deixe a área do acampamento mais limpa do que como você a encontrou.

A low-angle, upward-looking photograph of two modern skyscrapers with glass facades. The buildings are angled towards the right, creating a sense of height and scale. The sky is a uniform, overcast grey. The word "Resumo" is centered in the middle of the image in a large, bold, black sans-serif font.

Resumo

- Lembre-se: Late equals never!
- Defenda os interesses do código frente às demandas de projeto
- Escreva código para ser lido
 - Coloque-se no papel de outro dev lendo seu código
- Regra de escoteiro

Dúvidas e discussões

grupo ● 3778