

Boas Práticas em Programação

Miguel Alexandre Uhlein
Análise e Desenvolvimento de Sistemas
Faculdade SENAI Porto Alegre
Porto Alegre, Brasil
nedcloar1@hotmail.com

Gabriel Freitas Brasil
Análise e Desenvolvimento de Sistemas
Faculdade SENAI Porto Alegre
Porto Alegre, Brasil
gabriel.f.wlt@gmail.com

Resumo—Muitos programadores investem muito tempo e esforço na criação de seus códigos de programação. No entanto, muitos deles acabam se esquecendo de que, ao longo do tempo, os códigos atuais podem se tornar obsoletos, o que muitos chamam de código legado, que acabam exigindo manutenção ou até mesmo a reescrita. Este problema surge quando o programador não dá a devida atenção às boas práticas dentro do desenvolvimento de códigos, o que resulta em diversas linhas de código complexo, que acaba dificultando sua compreensão por outros membros da equipe. Neste artigo abordaremos a importância da clareza, da modularização, da documentação, simplicidade e da adoção de padrões consistentes de escrita do código para facilitar a manutenção, escalabilidade e colaboração ao longo do ciclo de vida do Software. Além disso, abordaremos como essas práticas não apenas melhoram a qualidade do código, mas também que impulsionam a produtividade durante seu desenvolvimento e para suas possíveis atualizações.

Palavras-chave: código, desenvolvimento, práticas, qualidade, programação

I. INTRODUÇÃO

Quando um desenvolvedor está começando algum projeto normalmente ele investe muito tempo e esforço durante o desenvolvimento, porém muitas vezes acabam se esquecendo que o código desenvolvido por ele pode em algum momento precisar de manutenção ou aprimoramento e acaba não seguindo boas práticas dentro do desenvolvimento de códigos como os seguintes princípios *Clean Code* ou Código Limpo, KISS (*Keep It Simple, Stupid*) Mantenha Isso Simples, Estupido, DRY (*Don't Repeat Yourself*) Não se Repita. No entanto essas práticas não são apenas sugestões de como melhorar ou estilizar o código e sim são consideradas por muitos desenvolvedores como pilares fundamentais para o desenvolvimento de software sustentável a longo prazo.

II. REFERENCIAL TEÓRICO

Durante a pesquisa para a elaboração deste artigo os autores consultaram blogs, podcasts sobre os assuntos abordados ao decorrer do artigo e aqui podemos ver relatos de outros desenvolvedores de software falando sobre o assunto

[...] Mantenha as coisas simples! Este conceito vem até de outro livro, e particularmente acho que é a base de uma boa solução. Normalmente

tendemos a complicar as coisas que poderiam ser muito mais simples. Então, *Keep It Stupid Simple* (Mantenha isto estupidamente simples - KISS)! (BALTIERI, ano não informado, Balta.io, Acesso em 14/11/2023)

[...] O que é código legível? Isto é subjetivo para algumas pessoas código legível é diferente e pra outras de acordo com suas próprias experiências, e a primeira página do primeiro capítulo do *clean code* é uma *chart* que ficou muito famosa que é qual é a medida mais eficaz de qualidade de código? que é o número de *what the fucks* por minuto que você fala quanto está lendo o código (ARCOVERDE, 2023, Podcast da Alura, Acesso em 14/11/2023)

[...] Esse princípio pode ser traduzido como “não repita a si mesmo”. Essa expressão foi descrita pela primeira vez em um livro chamado *The Pragmatic Programmer* e se aplica a diversas áreas de desenvolvimento, como: Banco de Dados; Testes; Documentação; Codificação. O DRY diz que cada pedaço do conhecimento de um sistema deve ter uma representação única e ser totalmente livre de ambiguidades. Em outras palavras, define que não pode existir duas partes do programa que desempenhem a mesma função. (Conheça os 7 princípios do *Clean Code*, Hostgator 2020, Disponível em: <https://www.hostgator.com.br/blog/clean-code-o-que-e/>, Acesso em 14/11/2023)

III. PROCEDIMENTO METODOLÓGICO

O princípio Clean Code, ou código limpo, refere-se a um código legível onde qualquer programador possa entender ao ler, ser um código testável para que possa ser verificado se os cenários de execução do programa estão de acordo com o esperado, facilmente mantido para que possa ser fácil de

ser atualizado ou dar manutenção, dentro do Clean Code existem diversas regras a serem seguidas. Uma delas é a utilização de nomenclatura clara e intuitiva, essa regra é utilizada em diferentes situações em um programa, como atribuir nomes a variáveis, funções, classes, parâmetros e até mesmo nos nomes dos arquivos existentes no projeto, um exemplo é quando um programador está fazendo uma aplicação sobre o tempo é necessário ter uma variável que armazena o horário atual e alguns programadores acabam colocando o nome da variável como "t" que deixa o código mais difícil de se entender, um bom nome seria *currentTime*, pois o nome da variável é sempre essencial seguir a essência sobre o que ela se trata com isso abordamos outras regras que seriam elas: Utilize variáveis autoexplicativas, Escolha nomes descritivos para classes, variáveis e métodos. Outra regra fala sobre declarar as variáveis próximas ao seu uso, com isso quando outro programador for dar manutenção em seu código ele não irá se perder facilmente ao ler o código procurando onde a variável utilizada em tal função foi declarada. E por fim outra regra cujo os autores deste artigo acham importante ser seguido é não quebre a indentação, essa regra fala sobre como a formatação ajuda na hora da leitura do código. Abaixo estão alguns exemplos de coisas que podem ser consideradas ao realizar a formatação do código

- Separar os conceitos verticalmente;
- Declarar variáveis próximas ao seu uso;
- Manter as linhas curtas;
- Não alinhar o código pela horizontal;
- Utilizar de espaços em branco para associar ou desassociar coisas relacionadas ou não relacionadas;

Já o princípio KISS (*Keep It Simple, Stupid*) Mantenha isso Simples Estupido é mais visto dentro do desenvolvimento do UX/UI (User Experience/User Interface) e fala sobre manter tanto o código quanto a interface que o usuário irá visualizar de uma maneira simples e intuitiva. Muitas vezes o programador pensa em uma interface inovadora e acaba fazendo algo muito complexo para o usuário comum entender, sem contar que com um interface complexa muitas vezes vem um código mais complexo ainda de se entender.

Por fim o princípio DRY (*Don't Repeat Yourself*) Não se Repita é considerado pelos autores deste artigo como por muitos outros programadores como um dos mais importantes a ser seguido pois muitas vezes em projetos extensos acabam tendo diversas linhas de códigos duplicados que no fim de uma compilação do projeto acaba deixando o projeto mais pesado em questão de armazenamento e consumo de recursos da máquina que irá executar a aplicação. Dentro do princípio DRY é falado sobre agrupar essas linhas de código duplicado em um único arquivo contendo a funcionalidade que essas linhas trazem e assim implementar essa função onde haviam os códigos duplicados, com isso quando o código passar por algum tipo de manutenção não será necessário modificar em todos os lugares onde as linhas duplicadas estavam e sim só na função criada para eliminar essas duplicidades.

Tendo em vista que as boas práticas de *clean code* não são utilizadas, os problemas são diversos como a dificuldade de leitura e compreensão, isto pode trazer dificuldades para a manutenção, com um código sujo ou mais desorganizado, o tempo do desenvolvimento irá ser maior tendo em vista que os desenvolvedores irão gastar mais do seu tempo para tentar compreender e modificar o código, outro ponto a ser idealizado é a dificuldade de colaboração entre os desenvolvedores pois códigos sujos ficam mais complicados tendo em vista que não seguem um padrão, podendo também atingir uma dificuldade de escalabilidade do software, tornando-o menos flexível para adição de novos recursos ou acomodação de crescimento, em resumo o uso dos princípios apresentados neste artigo são fundamentais para garantir a qualidade e a manutenção do software e ignorar estas técnicas pode gerar diversos problemas técnicos e financeiros significativos ao projeto.

IV. APLICAÇÕES E RESULTADOS

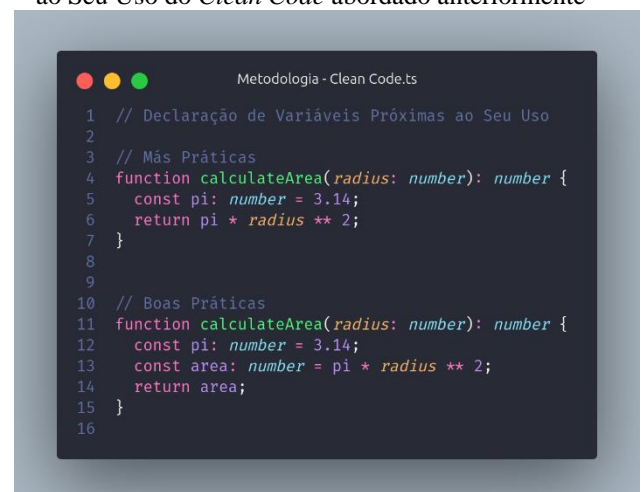
Figura referente à regra Utilização de Nomenclatura Clara e Intuitiva do *Clean Code* abordado anteriormente



```
Metodologia - Clean Code.ts
1 // Utilização de Nomenclatura Clara e Intuitiva
2
3 // Más Práticas
4 const t: Date = getT();
5 console.log(`t: ${t}`);
6
7 // Boas Práticas
8 const currentTime: Date = getCurrentTime();
9 console.log(`The current time is: ${currentTime}`);
10
11 /*
12 * OBS: Ambas funções getT e getCurrentTime
13 * Não existem no escopo do TypeScript.
14 * E foram criadas apenas como exemplo
15 */
```

Fonte: Autores

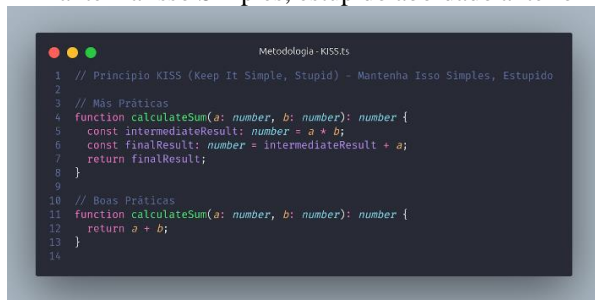
Figura referente à regra Declaração de variáveis Próximas ao Seu Uso do *Clean Code* abordado anteriormente



```
Metodologia - Clean Code.ts
1 // Declaração de Variáveis Próximas ao Seu Uso
2
3 // Más Práticas
4 function calculateArea(radius: number): number {
5     const pi: number = 3.14;
6     return pi * radius ** 2;
7 }
8
9
10 // Boas Práticas
11 function calculateArea(radius: number): number {
12     const pi: number = 3.14;
13     const area: number = pi * radius ** 2;
14     return area;
15 }
16
```

Fonte: Autores

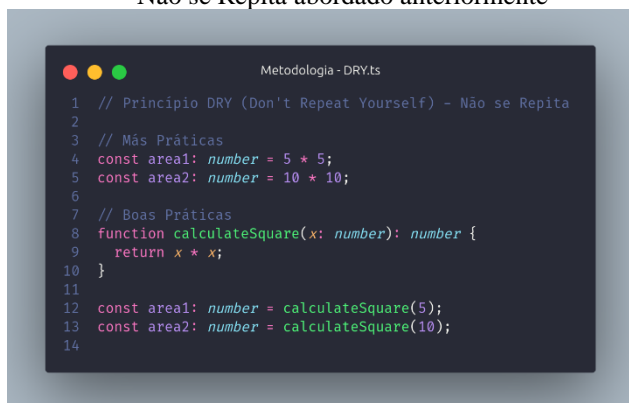
Figura referente ao Princípio KISS (*Keep It Simple, Stupid*)
– Mantenha Isso Simples, estúpido abordado anteriormente



```
1 // Princípio KISS (Keep It Simple, Stupid) - Mantenha Isso Simples, Estúpido
2
3 // Más Práticas
4 function calculateSum(a: number, b: number): number {
5   const intermediateResult: number = a * b;
6   const finalResult: number = intermediateResult + a;
7   return finalResult;
8 }
9
10 // Boas Práticas
11 function calculateSum(a: number, b: number): number {
12   return a + b;
13 }
14
```

Fonte: Autores

Figura referente ao Princípio DRY (*Don't Repeat Yourself*)
– Não se Repita abordado anteriormente



```
1 // Princípio DRY (Don't Repeat Yourself) - Não se Repita
2
3 // Más Práticas
4 const area1: number = 5 * 5;
5 const area2: number = 10 * 10;
6
7 // Boas Práticas
8 function calculateSquare(x: number): number {
9   return x * x;
10 }
11
12 const area1: number = calculateSquare(5);
13 const area2: number = calculateSquare(10);
14
```

Fonte: Autores

V. CONCLUSÃO

Concluímos que para qualquer pessoa que cria e reajusta códigos, necessidade de ter cada parte do projeto desenvolvido bem escrito, com as funções claras e corretas, pois todos que veem aquele código precisam entender o que é necessário estar escrito e o que está sendo realizado, em alguns momentos escutamos muito sobre comentar o código, algo que de fato pode ajudar no início, mas quando você tem um projeto muito extenso isso pode atrapalhar um pouco, o mais correto seria utilizar das práticas de *Clean Code* e criar as funções da maneira mais clara e objetiva possível pois este código pode ser a base de uma atualização no futuro do próprio desenvolvedor ou de outro profissional. O intuito deste artigo é orientar e mostrar a importância de termos um código em um padrão organizado e limpo.

REFERÊNCIAS

- [1] Furtado, Fernando. "*Clean Code*: O que é, Casos de Uso, Exemplo de Código Limpo." Alura, 24 abr. 2019. Disponível em: <https://www.alura.com.br/artigos/o-que-e-clean-code>. Acesso em: 14/11/2023.
- [2] Autor não informado. "*Clean code*: o que é, porque usar e principais regras!" Blog da Trybe, 19 fev. 2021. Atualizado em 10 dez. 2022. Disponível em: <https://blog.betrybe.com/tecnologia/clean-code/>. Acesso em: 14/11/2023.
- [3] Awari. "*SOLID, KISS, DRY, YAGNI*: Desvendando esses princípios de código." Awari Blog, 4 jan. 2023. Disponível em: https://awari.com.br/solid-kiss-dry-yagni/?utm_source=blog&utm_campaign=projeto+blog&utm_medium=SOLID,%20KISS,%20DRY,%20YAGNI,%20Desvendando%20esses%20princ%C3%AADpios%20de%20c%C3%B3digo. Acesso em: 14/11/2023.
- [4] Beltieri, André. "*Clean Code* - Guia e Exemplos." Balta Blog. Disponível em: <https://balta.io/blog/clean-code>. Acesso em: 14/11/2023.
- [5] Kriger, Brunno. "Entenda e descubra quais são as boas práticas de programação." Blog Kenzie, 19 jan. 2023. Disponível em: <https://kenzie.com.br/blog/boas-praticas-programacao/>. Acesso em: 14/11/2023.
- [6] SYDLE. "Clean code: quais são as boas práticas de programação?" Blog SYDLE, 18 mar. 2022. Disponível em: <https://www.sydle.com.br/blog/clean-code-602bef23da4d09680935509b>. Acesso em: 14/11/2023.
- [7] Alura. "Clean Code: O que é um código limpo?" YouTube, 20 out. 2023. Disponível em: <https://www.youtube.com/watch?v=FSqk2MKpVnI>. Acesso em: 14/11/2023.
- [8] HostGator. "Conheça os 7 princípios do *Clean Code*." Blog HostGator, 08 abr. 2020. Disponível em: <https://www.hostgator.com.br/blog/clean-code-o-que-e/>. Acesso em: 14/11/2023.