Regras de código limpo e padrões de indentação

Um bom código deve seguir boas regras e boas práticas de codificação. Para isso, estudamos documentos, usamos exemplos e padronizamos as regras de código do nosso projeto - a fim de organizar, aumentar a legibilidade e a manutenibilidade do nosso código.

1. **Boas práticas de UI e UX design**
   Muitos imaginam que a prototipação não entra nas regras e padrões de legibilidade, mas estão muito enganados. No nosso projeto, procuramos indentar corretamente o arquivo com as interfaces, a fim de deixar claro na hora de desenvolver o produto, quais são as tarefas a ser seguidas, os próximos passos e facilitar a busca e a pesquisa de informação ao codificar o projeto.

2. **Princípio de responsabilidade única**
   Cada função do código do nosso trabalho deverá ser responsável por executar apenas uma ação, criando um fator de organização melhor para o nosso código, e delegando tarefas secundárias para outras funções.

3. **Open/close principle**
   Objetos e entidades do nosso projeto devem ser abertos para extensões, porém não para modificações, ou seja, nossas funções iniciais continuarão fazendo o que faziam no começo do desenvolvimento do produto, porém para que haja mais funcionalidades, possivelmente, será necessário fazer extensões nas nossas funções, sem que haja modificações, trazendo uma maior facilidade de manutenção para o código, não trazendo a necessidade de alterar várias funções devido a alteração de apenas uma.

4. **Regras de nome (variáveis, funções, classes, etc)**
   Nomes de variáveis e funções devem ser simples, fáceis de entender e de ler. Assim, procuramos sempre usar nomes expressivos e que façam sentido e se adequem à realidade do nosso projeto.

   Por exemplo:

```
// Evitar nomes confusos e que não tenham métrica
var x = 25;
// Duração do que? Qual a métrica?
int duration = 25;
// Muito mais expressivo e simples de entender!
int durationInMinutes = 25;
```

5. **Princípio da inversão de dependência**
   O princípio da inversão de dependência consiste na inversão do senso comum de que deveríamos basear a perspectiva do avanço do projeto no código, sendo que iremos pensar primeiramente na abstração do nosso projeto, então iremos basear o código para elucidar essa abstração, e não o contrário.

6. **Regras de comentários**
   Um bom uso de comentários e padrões, é sobre descrever certas variáveis, funções e métodos - a fim de evitar tempo de leitura e entendimento sobre aquelas linhas de código. Sendo assim, utilizamos comentários expressivos, simples, diretos e objetivos, que descrevem muito bem o que determinada função faz ou reproduz.

   Por exemplo:

```
// Retorna a lista de produtos
// para o relatório de fechamento mensal
public List<Product> ObtemProdutos()
{
    ...
}
```