
mundipagg

Thiago Barradas

Software Enginner | Mundipagg

[Web Applications] [ASP .NET]
[API RESTful] [Microsoft ♥ Linux]
[Elasticsearch] [Docker]
[DevOps] [Agile]

tbarradas@mundipagg.com
LinkedIn: thiagobarradas
(21) 99329-9143

CLEAN CODE

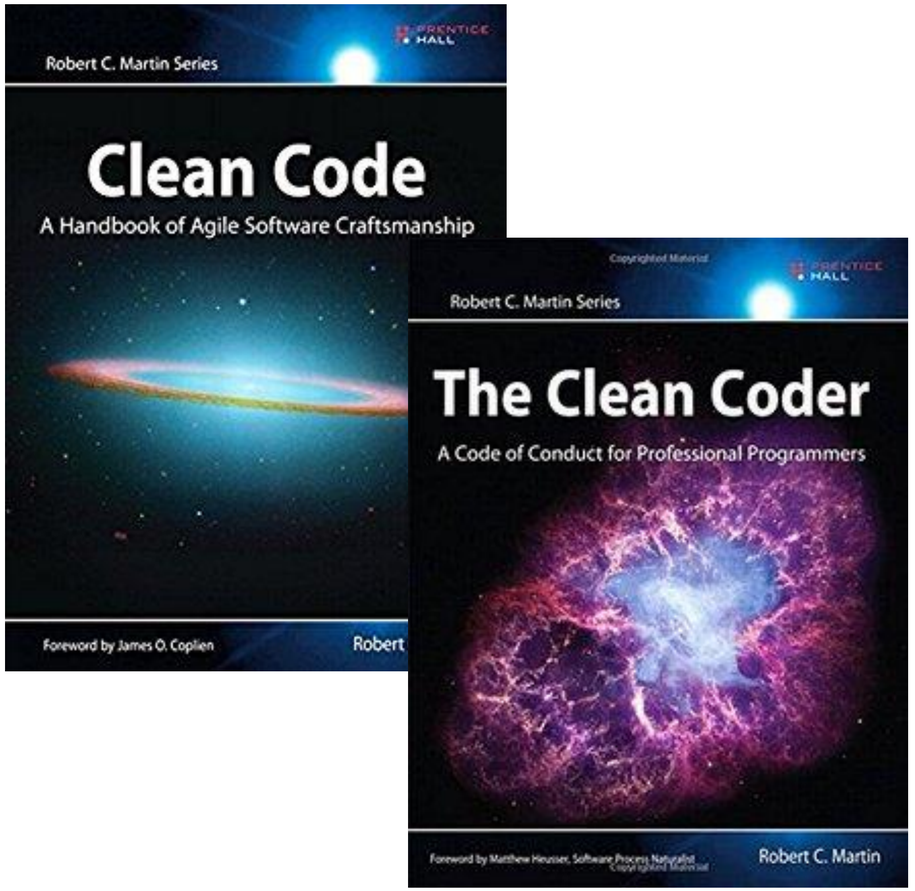
POR UM MUNDO COM CÓDIGOS MELHORES

POR QUE

VOCÊ ESTÁ AQUI?

PRECISAMOS MUITO DE UM MUNDO COM CÓDIGOS MELHORES

The Clean Coder



Robert C. Martin
(Uncle Bob)

Princípios do Clean Code

- Eficiente
- Simples
- Direto ao ponto
- Fácil leitura
- Sem dependências
- Fácil manutenção
- Padrões definidos
- Sem duplicação
- Coberto de testes
- Elegante

*"**Qualquer tolo** consegue escrever código que um computador entende. **Bons programadores** escrevem código que humanos podem entender."*

Martin Fowler

POSSO VER SEU CÓDIGO?


4GIFs.com

“Ainda ta bagunçado, depois vou melhorar e refatorar o código.”

FILHO FEIO

NÃO TEM PAI!

“Fiz o meu melhor dentro do tempo que eu tinha.”

“Mas o prazo era extremamente curto.”

“Meu gestor me faz pressão o tempo inteiro.”

“Preciso entregar, preciso ser muito produtivo.”

NÃO EXISTE DESCULPA

PARA UM CÓDIGO RUIM!

TENHA ORGULHO DO SEU CÓDIGO!

OK. CHEGA DE LERO-LERO,

VAMOS AO QUE INTERESSA!

#1

NOMENCLATURA

Nomenclatura

- Use nomes que revelem a intenção;
- Use nomes pronunciáveis;
- Não importa que o nome seja grande;
- Mantenha o padrão, sempre;

Nomenclatura

```
int txI;
repo.Get();
```

```
int transactionInstallments;
transactionRepository.GetTransactions();
```

#2

Classes

Classes

- Ser representadas por substantivos;
- Nome claro ao seu contexto;
- Não importa que o nome seja grande;
- Mantenha o padrão, sempre;

Classes

```
public class Cut
{
    //...
}

public class CustomerDTO
{
    //...
}
```


```
public class Cutter
{
    //...
}

public class CustomerRepository
{
    //...
}
```

#3

Métodos

Métodos

- Devem ser representados por verbos;
- Nome claro ao seu contexto;
- Devem ser pequenos. Quanto menor, melhor;
- Extraia sempre que puder em métodos auxiliares, seja um método privado ou utilitários;

Métodos

- Um método deve fazer apenas uma coisa. Uma;
- Argumentos booleanos não costumam ser elegantes.
- Muitos parâmetros devem ser substituídos por objetos.
- Não importa que o nome seja grande;
- Mantenha o padrão, sempre;

Métodos

```
public void New(string name, string email, int age)
{
    var repo = new UserDTO();
    var u = repo.Get(email);
    if (u == null)
    {
        //...
    }
}

public List<Transaction> GetTransactions(bool isAuthorized)
{
    //...
}
```

Métodos


```
public void RegisterUser(User user)
{
    if (IsEmailAvailable(user.Email))
    {
        //...
    }
}

public List<Transaction> GetAuthorizedTransactions()
{
    //...
}

public List<Transaction> GetNotAuthorizedTransactions()
{
    //...
}
```

#4

Condições

Condições

- Sempre que possível, extrai para métodos privados;
- Não importa que o nome seja grande;
- Mantenha o padrão, sempre;

Condições

```
if (transaction.Status == "Authorized" || transaction.Status == "Captured")
{
    //...
}
```


```
if (IsEligibleForCancellation(transaction))
{
    //...
}

private bool IsEligibleForCancellation(Transaction transaction)
{
    return (transaction.Status == TransacationStatusEnum.Authorized ||
            transaction.Status == TransacationStatusEnum.Captured);
}
```

#5

Objetos e Estruturas de Dados

Objetos e Estruturas de Dados

Basicamente devem seguir a **Lei de Demeter**:

Um método **M** de uma classe **C** só deve conhecer:

- **Métodos de C;**
- **Objetos criados por M;**
- **Objetos passados por parâmetro para M;**
- **Objetos em propriedades de instâncias de C.**

#6

Abstração

Abstração

- Tente ao máximo generalizar suas classes;
- Abstraia em quantos níveis for preciso;
- Não importa que o nome seja grande;
- Mantenha o padrão, sempre;

Abstração

```
public class CreditCardTransaction
{
    public long AmountInCents { get; set; }

    //...
}

public class OnlineDebitTransaction
{
    public long AmountInCents { get; set; }

    //...
}
```

Abstração


```
public abstract class Transaction
{
    public long AmountInCents { get; set; }
}

public class CreditCardTransaction : Transaction
{
    //...
}

public class OnlineDebitTransaction : Transaction
{
    //...
}
```

#7

Comentários

Comentários

- Um código bem escrito dispensa comentários;
- São aceitáveis quando há necessidade de explicação do negócio
- Também para licença de uso e documentação;
- Não importa que o comentário seja grande;
- Mantenha o padrão, sempre;

Comentários

```
// check if gateway is MundiPagg
if (gateway == "MundiPagg")
{
    transaction.Retry();
}
```


```
// MundiPagg is the only gateway that retries a transaction
if (gateway == "MundiPagg")
{
    transaction.Retry();
}
```

#8

Formatação do Código

Formatação do Código

- Não existe um padrão definitivo para formatação;
- Defina regras no time;
- Tente sempre se basear no padrão que a comunidade propõe;
- Não importa que o padrão seja grande;
- Mantenha o padrão, sempre;

#9

Exceções

Exceções

- Utilize exceções específicas para seus erros;
- Evite códigos de erros;
- Trate exceções exclusivamente em métodos;
- Não importa que o tratamento seja grande;
- Mantenha o padrão, sempre;

Exceções

```
TransactionResult result = GetTransaction(transactionId);
if (result.Success == false && result.Error == Error.NOT_FOUND)
{
    //...
}
return result.Transaction;
```

Exceções


```
try
{
    return GetTransaction(transactionId);
}
catch (NotFoundException ex)
{
    //...
}
catch (Exception ex)
{
    //...
}
```

#10

Testes

Testes

- USE o TDD;
- Se não tiver, implemente testes continuamente a cada implementação/correção;
- Use e abuse; Teste de integração, funcional, etc;
- Mantenha o padrão, sempre;

Testes

Podemos seguir também o “**F.I.R.S.T.**”:

Fast: Testes rápidos, resultados rápidos.;

Independent: Não deve ter dependências;

Repeatable: Os testes devem funcionar em qualquer ambiente;

Self-validation: Não é necessária nenhuma validação manual;

Timely: Os testes devem ser elaborados antes do código;

#11

Regra do Escoteiro

Regra do Escoteiro

“Deixe a área do acampamento mais limpa do que quando e como você a encontrou.”

Regra do Escoteiro

“Deixe o código mais limpo do que quando e como você o encontrou.”

Thiago Barradas

tbarradas@mundipagg.com

+55 (21) 99329-9143

LinkedIn: thiagobarradas

Obrigado!