

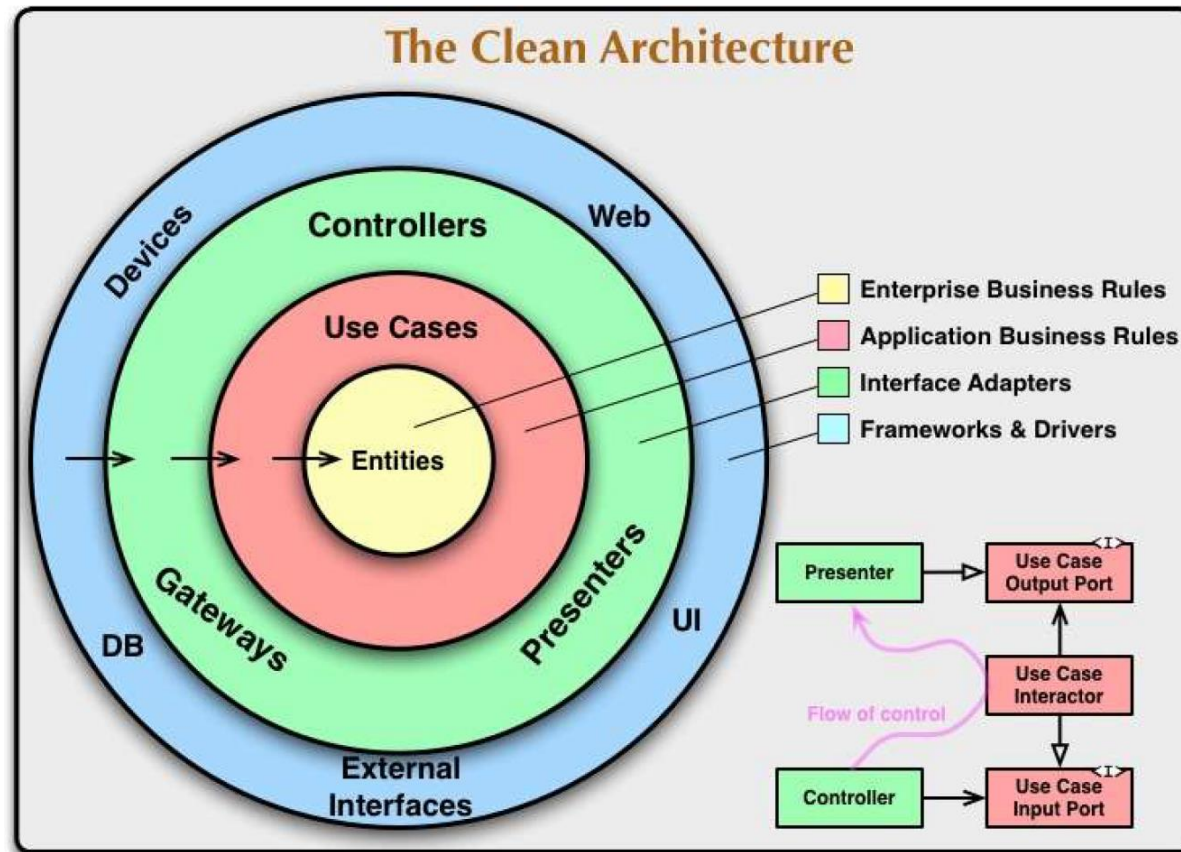


Boas Práticas de Desenvolvimento

Hader Azzini
Sr Tech Lead Data Science



Arquitetura Limpa



Testável

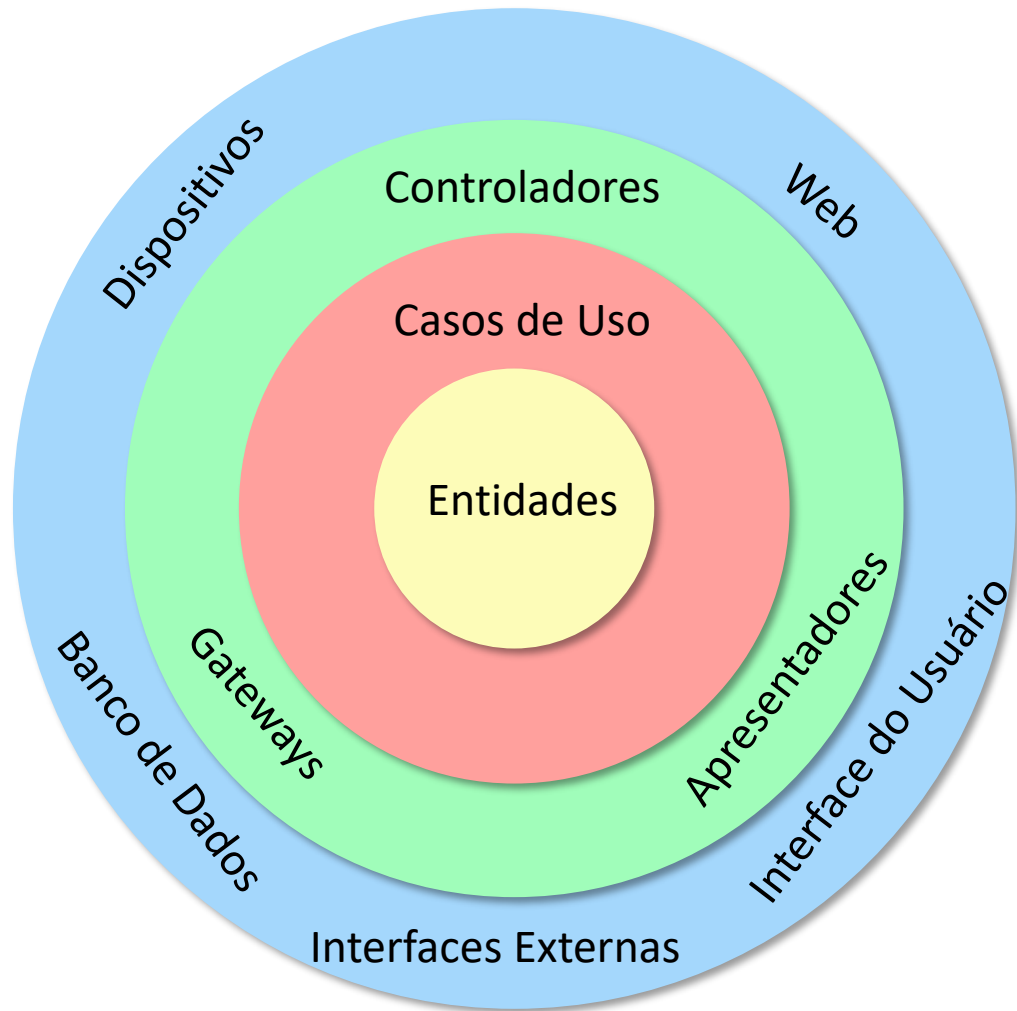
Independente de:

- Frameworks
- Interface do Usuário
- Banco de Dados
- Ações Externas

<http://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>

AS PARTES BÁSICAS DE UMA ARQUITETURA LIMPA

Analogia com uma Fábrica




Entidades

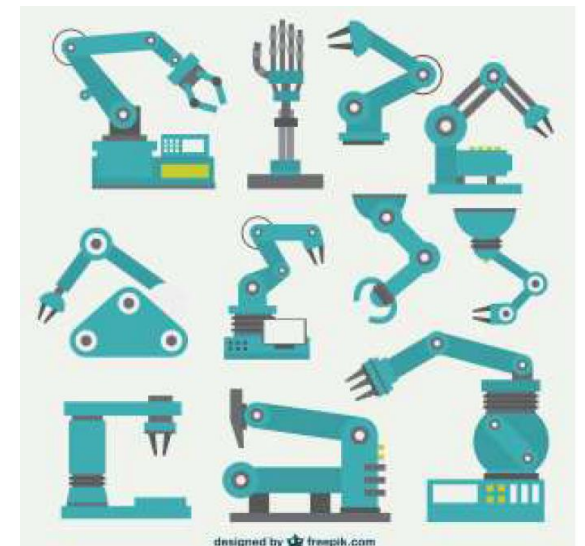
Objeto com métodos um conjunto de estruturas e funções de dados

Encapsulam as regras mais gerais e de alto nível.

Nenhuma mudança operacional deve afetar a camada da entidade.



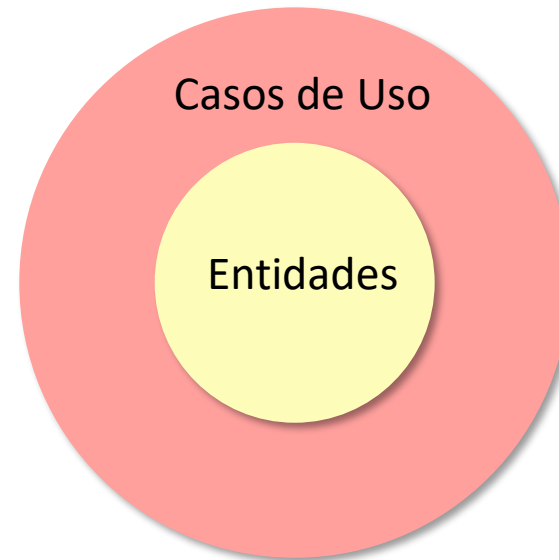
Entidades



Casos de Uso

Orquestram o **fluxo de dados** de e para as entidades e **direcionam** essas entidades.

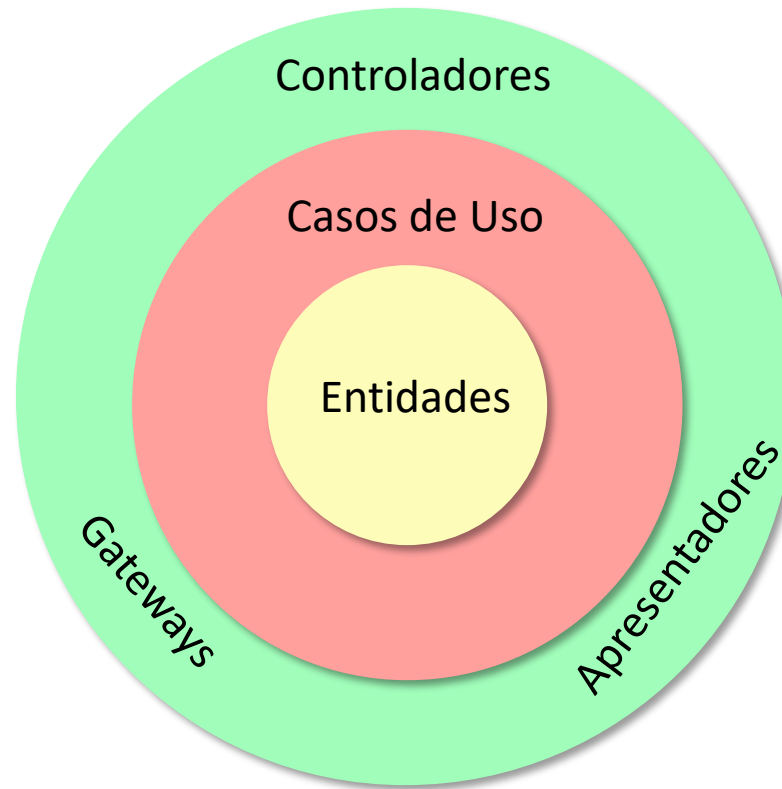
Apenas mudanças nas entidades podem afetar os casos de uso



Adaptadores de Interface

Os dados são convertidos, nessa camada, da forma **mais conveniente** para entidades e casos de uso.

Também na forma mais conveniente para banco de dados.

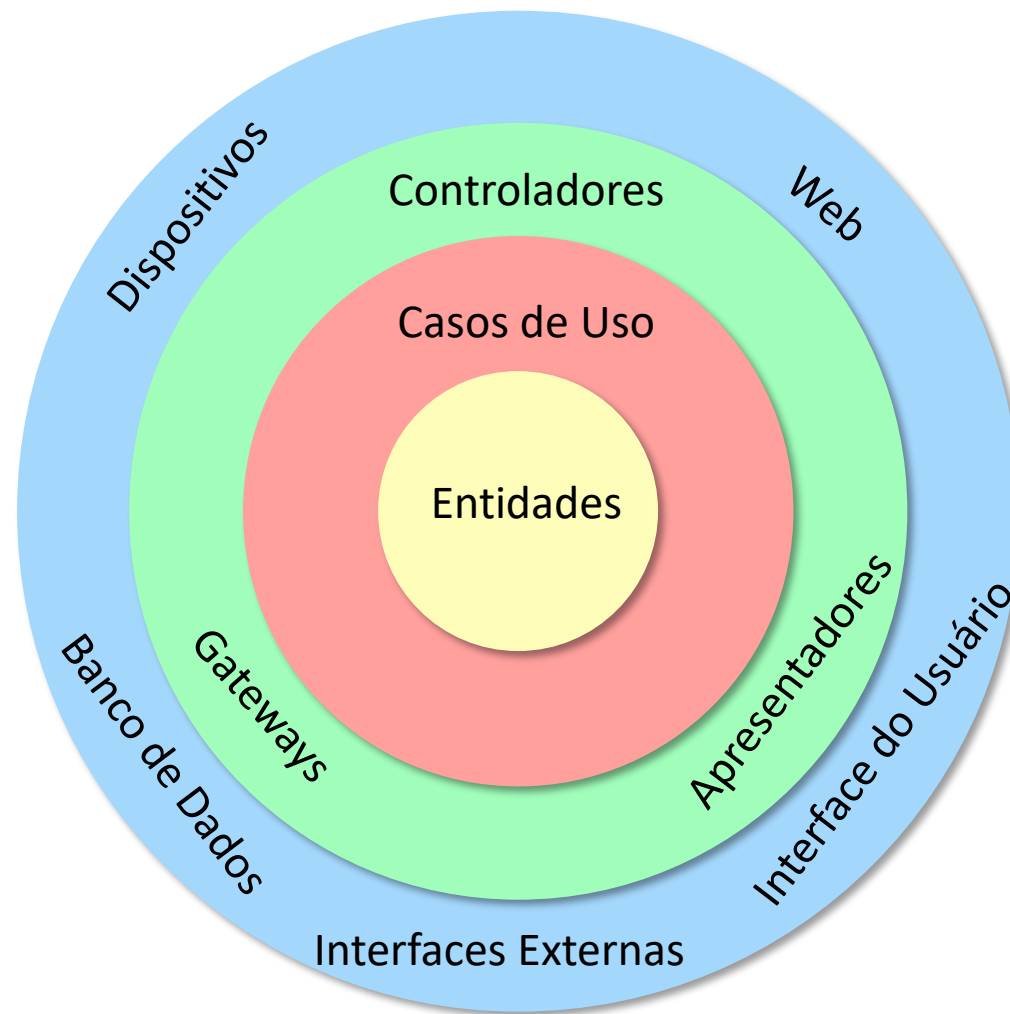


Interfaces Externas

É composta de estruturas e ferramentas, como o Banco de Dados, o Web Framework, etc.

Nada além do código que “conecta” o mundo externo ao seu programa.

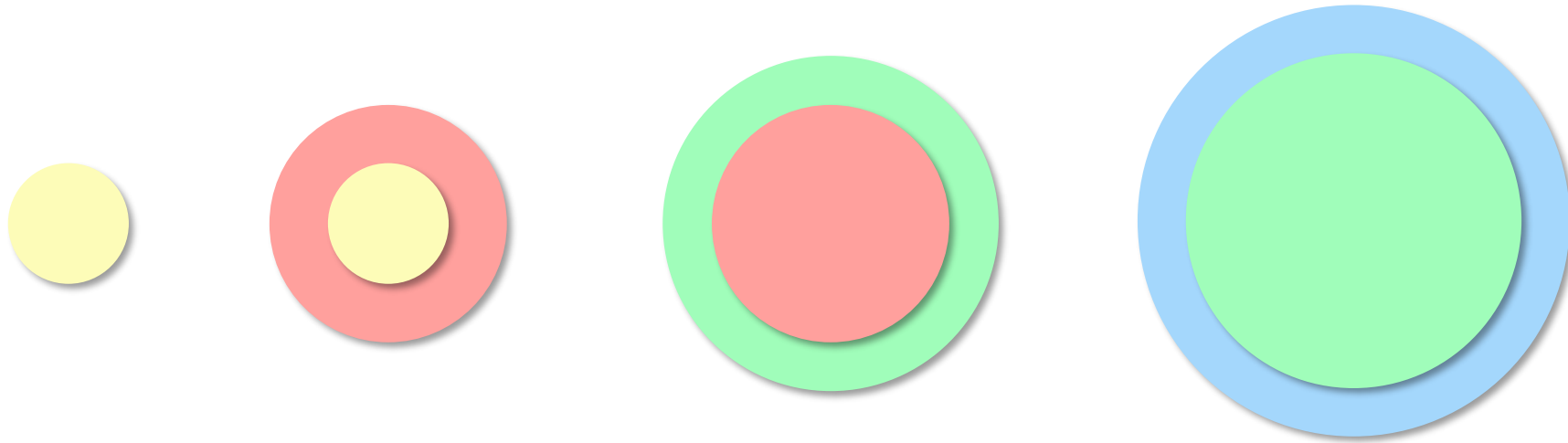
Esta camada é onde todos os detalhes vão. (Web, Banco de dados etc..)



AS REGRAS DE UMA ARQUITETURA LIMPA

Regra da Dependência

Nada em um círculo interno pode saber alguma coisa sobre algo em um círculo externo



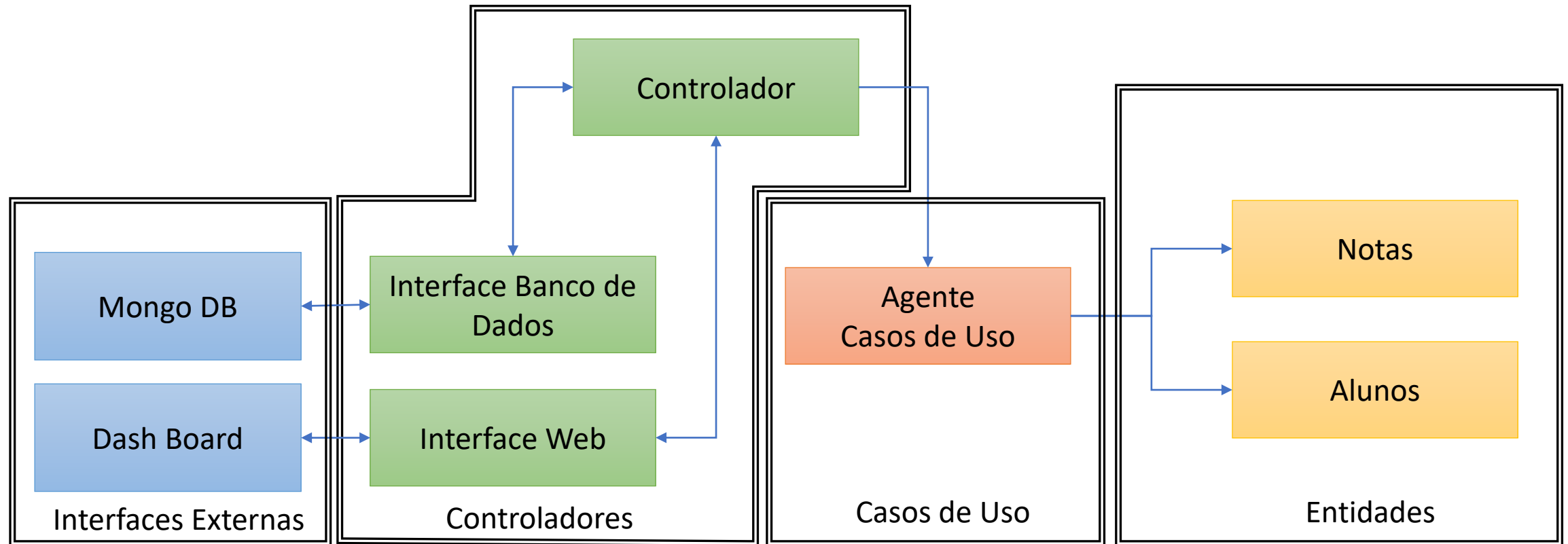
Apenas Dados Simples Cruzam as Fronteiras

Quando passamos dados através de uma fronteira, é sempre na forma que é mais conveniente para o círculo interno.

Não queremos que as estruturas de dados tenham qualquer tipo de dependência que viole a Regra de Dependência.

UM **EXEMPLO** DE UMA ARQUITETURA LIMPA

Software de Notas de Alunos



DESENVOLVIMENTO ORIENTADO A TESTES

Primeiro o teste depois o programa

Entrada

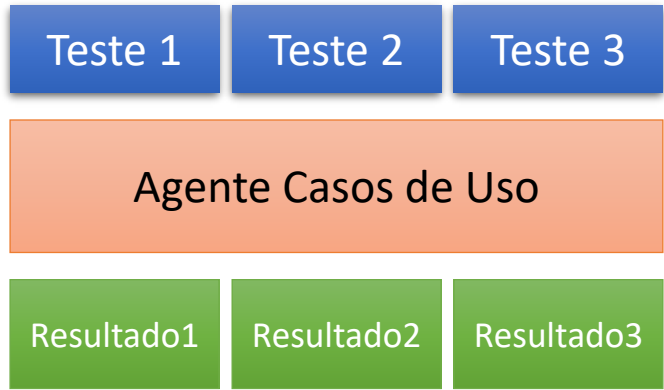
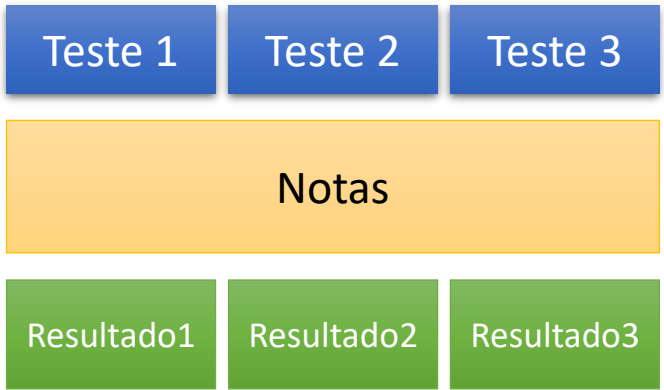
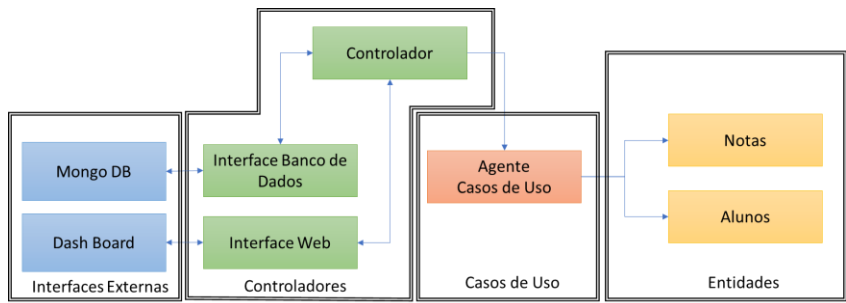
Resultado
Esperado

Entrada

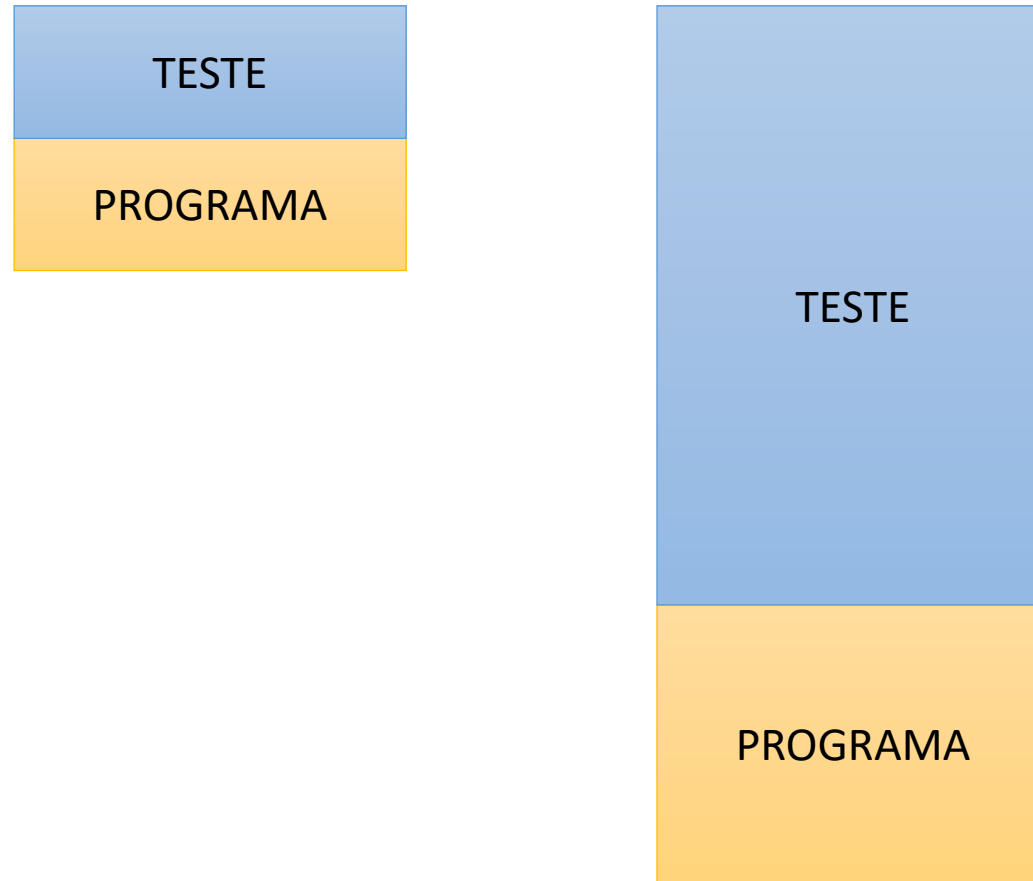
PROGRAMA

Resultado
Esperado

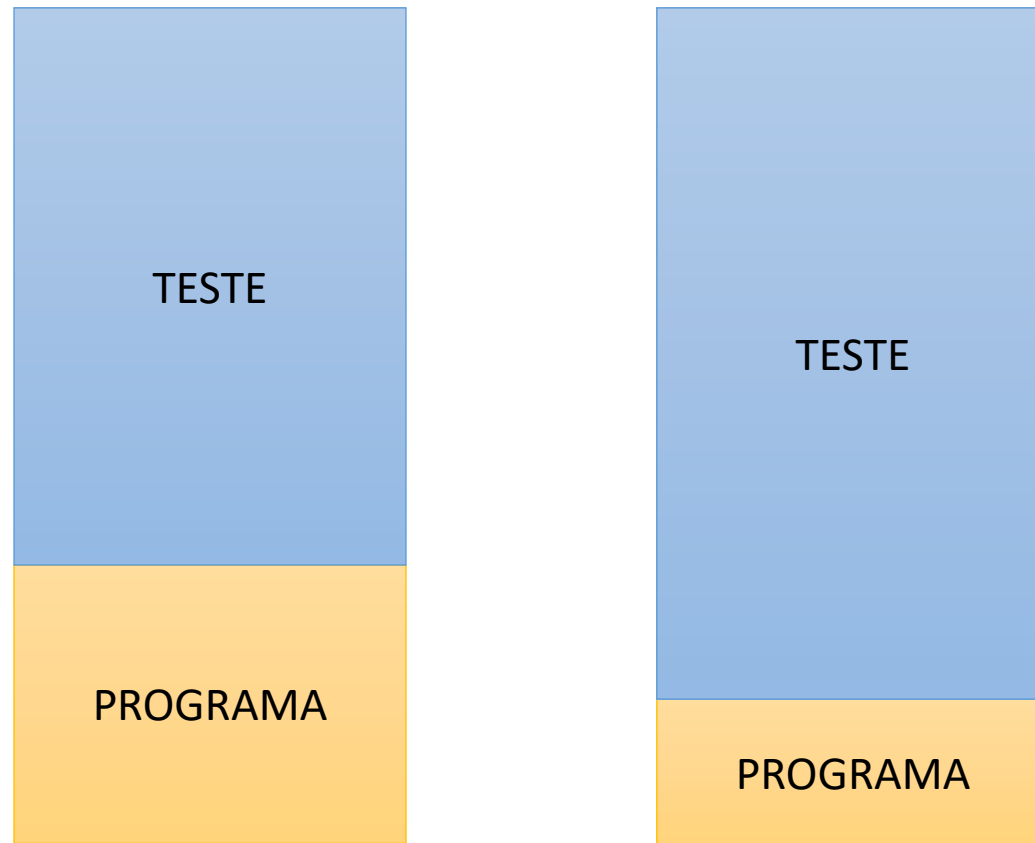
Teste cada coisa separadamente



Mais Teste menos Programa



O programa não está pronto quando faz o que deveria...



...mas quando faz o que deveria de forma simples.

S.O.L.I.D.

**OS CINCO PRINCÍPIOS DE
PROGRAMAÇÃO
ORIENTADA À OBJETO**

Meet up - Boas Práticas de Desenvolvimento

...

Pontos: 0/2

1

Escolha a hora de responder cada pergunta de acordo com a apresentação
(1 Ponto)

- ☐ Pergunta 1
- ☒ Pergunta 2
- ☐ Pergunta 3
- ☐ Pergunta 4
- ☐ Pergunta 5

< Slide anterior

Próximo slide >

Formulário



Qual é a boa Prática?

A)



```
class forma_geometrica():  
    def circulo()  
  
    def retangulo()  
  
    def triangulo()
```


B)



```
class circulo()  
  
class retangulo()  
  
class triangulo()
```

1 . Single-responsibility Principle

- Cada classe deve ter apenas uma responsabilidade



```
class forma_geometrica()  
    def circulo()  
  
    def retangulo()  
  
    def triangulo()
```



```
class circulo()
```

```
class retangulo()
```

```
class triangulo()
```

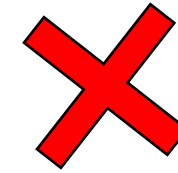

Qual é a boa Prática?

A)



```
class retangulo():  
    def calcula_area()  
  
    def calcula_area_velha()
```

B)



```
class retangulo():  
    def calcula_area()  
        ....(alterando aqui)
```

2 . Open-closed Principle

- Objetos e entidades devem estar abertos para extensão mas fechados para alterações



```
class retangulo():  
    def calcula_area()  
  
    def calcula_area_velha()
```



```
class retangulo():  
    def calcula_area()  
        ....(alterando aqui)
```

Qual é a boa Prática?

A)



```
class mamifero():  
    def move()  
    def respira()
```

```
class cachorro(mamifero):  
    def late()
```

```
class gato(mamifero):  
    def mia()
```

B)



```
class mamifero():  
    def move()  
    def respira()  
    def emite_som()
```

```
class cachorro(mamifero):
```

```
class gato(mamifero):
```

3. Liskov substitution principle

- Objetos devem ser substituíveis com instâncias de seus tipos base.

```
class mamifero():  
    def move()  
    def respira()
```



```
class cachorro(mamifero):  
    def late()
```

```
class gato(mamifero):  
    def mia()
```

```
class mamifero():  
    def move()  
    def respira()  
    def emite_som()
```



```
class cachorro(mamifero):
```

```
class gato(mamifero):
```


Qual é a boa Prática?

A)



```
class forma_geometrica():  
    def retorna_valores()  
        return area,perimetro
```

B)



```
class forma_geometrica():  
    def area()  
        return area  
    def perimetro()  
        return perimetro
```

4. Interface segregation principle

- Muitas interfaces específicas são melhores que uma interface de uso geral

`class forma_geometrica():`
 `def retorna_tudo()`
 `return area,perimetro`



`class forma_geometrica():`
 `def area()`
 `return area`
 `def perimetro()`
 `return perimetro`



Qual é a boa Prática?

A)



```
class Login():
```

```
class Google(Login):
```

```
class Azure(Login):
```

```
class AWS(Login):
```

B)



```
class LoginGoogle():
```

```
class LoginAzure():
```

```
class LoginAWS():
```

5. Dependency Inversion principle

- Módulos de alto nível não devem depender de módulos de baixo nível. Ambos devem depender de abstrações.
- Abstrações não devem depender de detalhes. Detalhes devem depender de abstrações.

```
class Login():
```

```
class Google(Login):
```

```
class Azure(Login):
```

```
class AWS(Login):
```



```
class LoginGoogle():
```

```
class LoginAzure():
```

```
class LoginAWS():
```



CÓDIGO MAIS HUMANO

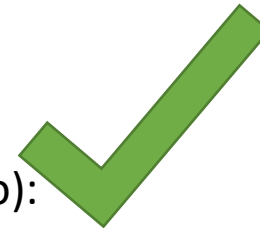
Facilite a Leitura

- Perdermos mais tempo lendo código do que programando
- “*spend ~58% of their time on program comprehension activities*” [1]

`def calc_a_cir(r):`



`def calcula_area_circulo(raio):`



[1] X. Xia, L. Bao, D. Lo, Z. Xing, A. E. Hassan and S. Li, "Measuring Program Comprehension: A Large-Scale Field Study with Professionals," in IEEE Transactions on Software Engineering, vol. 44, no. 10, pp. 951-976, 1 Oct. 2018, doi: 10.1109/TSE.2017.2734091.

Menos comentários

- Mais comentários = mais coisa para fazer manutenção
- Mas... “Deixe de comentar apenas se você escreve código compreensivo”

LEMBRE-SE

“ CÓDIGO = KARATÊ ”