Bons e Maus Testes

Para entender como realizar bons testes primeiro precisamos entender alguns conceitos de testes um dos principais conceitos são os pilares de testes que abordaremos em seguida:

Princípios de testes

Os testes tem 4 pilares básicos:

Proteção Contra Regressões

O primeiro pilar que iremos discutir é a o pilar de proteção contra regressões, já discutimos um pouco o que significa regressões, mas retomando é quando uma funcionalidade para de funcionar devido a um evento, muitas vezes a alteração de código; E este pilar fala especialmente sobre esse pilar como garantimos que o a cada nova modificação que realizarmos os testes que ja criamos seja o suficiente para avaliar que a funcionalidades as quais já tínhamos ainda se mantem, ou que caso não esteja a mesma seja apontada pelos testes.

Dentro dessa pilar nos temos que ter a ciência que quanto maior o código, bibliotecas internas ou externas, existira uma maior probabilidade de gerar uma regressão e bugs. devemos também avaliar a complexidade do código, no que diz respeito a regra de negocio e se as nossas asserções cobrem bem esses requisitos de negócios. de forma que exercitemos a maior quantidade de código possível. Entretanto é claro que existe um porém que e realização de testes triviais onde estes não tem muito valor, principalmente para esse pilar pois os mesmo não tem muita chance de ocasionar em regressões.

```
# exemplo de codigo trivial
def corta_nome_char20(nome:str) -> str:
  return nome[:20]
```

```
class Usuario:

  def __init__(self, nome):
    self._nome = nome

  # serua yn testes trivial testar
```

```
# um get simples como esse
def get_nome_user(self):
  return nome
```

Resistência á Refatoração

Primeiramente o que seria Refatoração?

significa realizar alteração em código já existente com o objetivo, diminuir a complexidade e melhorar legibilidade, sem alterar o comportamento da funcionalidades alteradas.

Agora podemos continuar, o pilar de resistência á refatoração, apresentar o grau a qual os testes suportam, as modificações das funcionalidades, sem que os testes quebrem, em geral os testes são feitos para que a cada modificação eles acusem o comportamento irregular pela funcionalidade, mesmo quando você não a quebre.

Uma das grandes características desse e garantir que principalmente que esses falsos positivos, não sejam capturados a cada refatoração. Quando alterado um funcionalidade e o mesmo mantem o comportamento intacto, e este não não são capturados pelos testes e denominado como uma testes unitário com resistência a refatoração.

E quais são os benefícios disso?, são essencialmente dois;

- Garantir que no caso de a funcionalidade seja quebra tenhamos um aviso preventivo, de um defeito e para que possamos corrigi-lo de maneira tempestiva, permitindo que não entre em produção
- E a confiança em relação as mudanças realizadas, poder confiar que as mudanças, realizadas estão muito menos propensas a gerar alguma regressão;

O Problema dos Falsos Positivos:

Um grande problemas dos falsos positivos, é quando eles se tornam parte da rotinas de teste, e é muito comum encontra-los, quando as pessoas se acostumam com as falhas, passam a ignora-las, também poderá acontecer de ignorar as falhas legitimas, apontadas pelos seus testes e isso pode se tornar muito, problema a curto ou a longo prazo. Um outra característicasdecorrente é o aumento da desconfiança em relação as mudanças e a hesitação e incerteza.

Manutenibilidade

Eventualmente avalia o custo de manutenção, primeiramente podemos apontar o tamanho do código de teste, o quão legível e simples para realizar mortificações, este código é. Porém não nos devemos nos preocupar em reduzir os nossos testes, pois isso afetaria de forma direta a qualidade dos Testes. Outro ponto é as dependência, em principal as foras do processo, o que e necessário para rodar os testes, conectividade, ter um banco de dados disponível, o quanto custa para manter essas dependências.

Feedback Rápido

Feedback rápido, tem uma caráter um tanto quanto descritivo, representa a velocidade que temos uma resposta dos teste, isso se reflete no tempo de execução dos teste, em principal nos testes de unidade e esperado que tenha um tempo menor que os outros.

Pesando os Testes

Entre os pilares de temos 2 deles são muitos importantes e geralmente esperamos máxima o grau de confiança neles, o primeiro é o de proteção contra e regressões e o segundo de de resistência a refutação

realização dos testes	Esperado Negativo	Esperado Positivo
Resposta Negativa	Negativo	False Negativo (falha proteção contra regressão)
Resposta Positiva	Falso positivo (falha na resistência a refatoração)	Positivo

Observando a tabela acima, podemos identificar que temos um relação entre a asserções dos testes e esses pilares, uma ponto importante e que, decisivamente é importante que primeiramente o código tenha uma proteção contra regressões, nas suas vazes iniciais enquanto, não houver ainda modificações e conforme for avançando no seu desenvolvimento também tenha um grau de resistência fatoração. Ou seja inicialmente os falsos positivos não são tão preocupantes quanto os falsos negativos, mas com o tempo pode se tornar um grande problema.

E uma da maneiras realizar o calculo do ruido gerados pelos testar, a precisão testes em relação ao ruido é realizando a diferença entre os bugs encontrados sobre os falsos alarmes

$$X = \frac{\sigma}{\epsilon}$$

σ = Quantidade de bugs encontrados

ε = Quantidade de falsos alarmes

X = Precisão dos testes em relação ao ruido

Estimando valor dos testes:

Para estimar os valores de testes devemos, utilizar os quatro pilares como parâmetros, para um testes ser valiosos, precisamos pontuar pelo menos alguma coisa em cada pilar, e para isso definimos uma grau de valor, entre 0 e 1, que o nosso destes pontua e o seu produto representa o valor estimado para o teste. Podemos representar pela seguinte equação:

$$valor = [0..1] * [0..1] * [0..1] * [0..1]$$

Entretanto eu não gosto muito bem dessa abordagem pois, basta uma métrica seja mais abaixo que a maioria e o valor do seu teste despenca e muitas vezes, para pessoas de negócios, que tem acompanhar não conseguiriam ver o real valor pelo resultado final. um dos fatores que contribui para eu não gostar dessa abordagem é que proteção contra regressões, resistência a refatoração e feedback rápido, são valores exclusivos então para potencializar dois deles e necessário defasar um outro. Isso eventualmente já seria um indicativo para o valor de testes decair. Para mim uma medida complementar que ajuda a visualizar melhor o equilíbrio entre os pilares, porém não exclui a necessidade do calculo do valor é a seguinte:

$$balançoteste = \frac{[0..1] + [0..1] + [0..1] + [0..1]}{4}$$

E apenas uma media simples entretanto nos permite ver o quanto nossos testes estão em equilíbrio em relação ao pilares de testes. Deve se pontuar que os sacrifícios em ralações aos testes devem apresentar uma redução nunca esperamos que os valores cheguem a zero. de forma que devemos sempre pontuar e tais reduções devem levar em consideração motivos estratégicos ou serem parciais. contribuindo para esse ponto, e necessário dizer que a resistência a refatoração e inegociável e deve tentar sempre conseguir performar melhor neste quesito.

Avaliando testes e pilares exclusivos

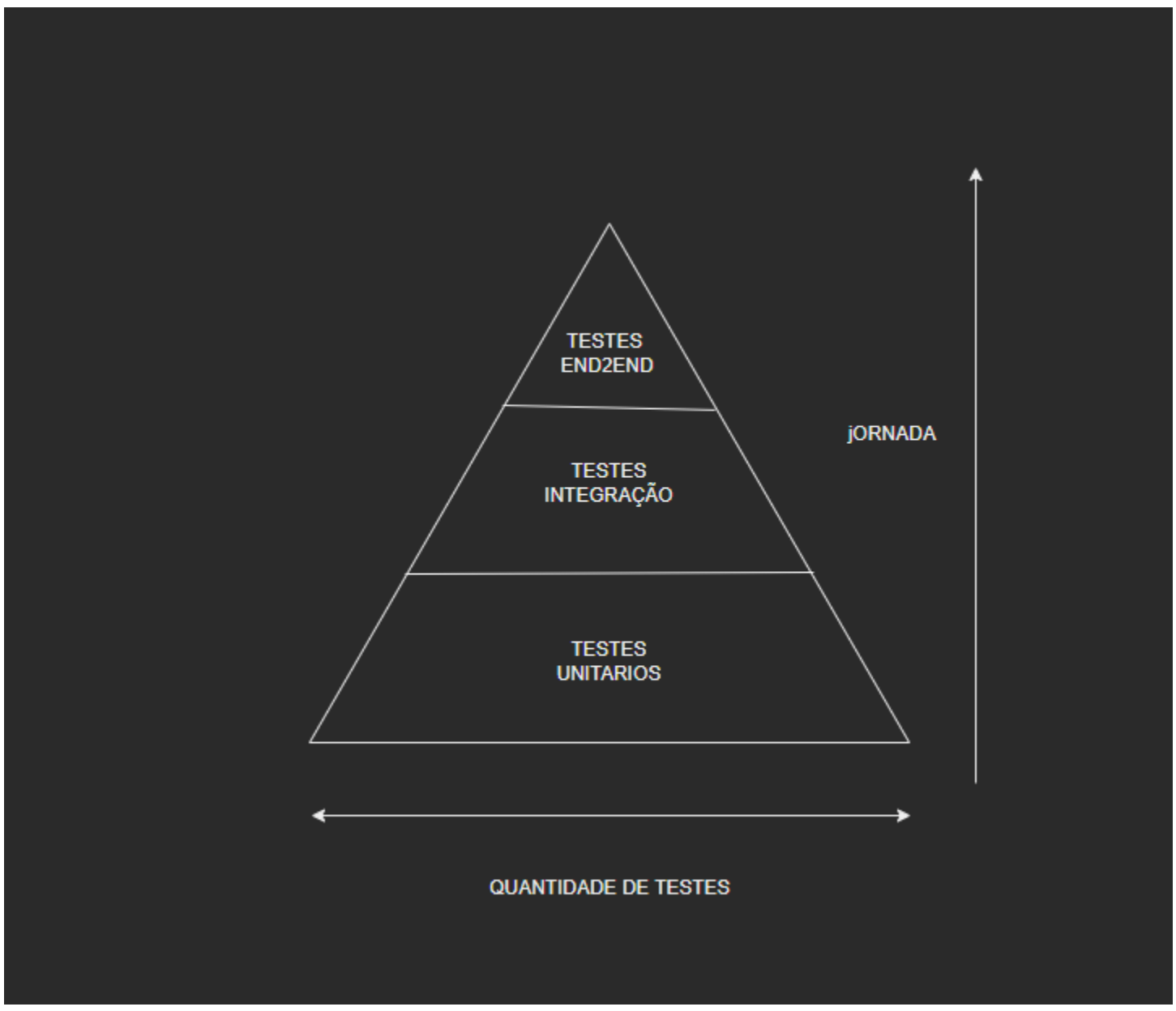
primeiramente vou mostra uma com alguns detalhes sobre testes e os 3 pilares, proteção contra regressões, resistência a refatoração e feedback rápido.

Testes	Resistencia a Refatoração	Proteção contra Regressões	Feedback Rapido
Testes Triviais	Alto - baixa chance de produzir falsos positivos	Inexistente - sem capacidade de revelar qualquer regressão	Alto - São rápidos
Testes Frágeis	Bem Baixo - Produz vários falsos positivos	Regular para Bom - tem chance de revelar regressões	Alto - executam rapidamente
Testes E2E	Alto - imune a falsos positivos	Alto - fornecem melhor proteção contra regressão	Bem Baixo - são lentos

Referente o pilar de manutenibilidade os testes triviais e testes frágeis, por serem mais fáceis também de escrever, porém os testes End-to-End, são mais robustos, e tem um custo maior em mantê-los.

Esclarecendo a Pirâmide de testes

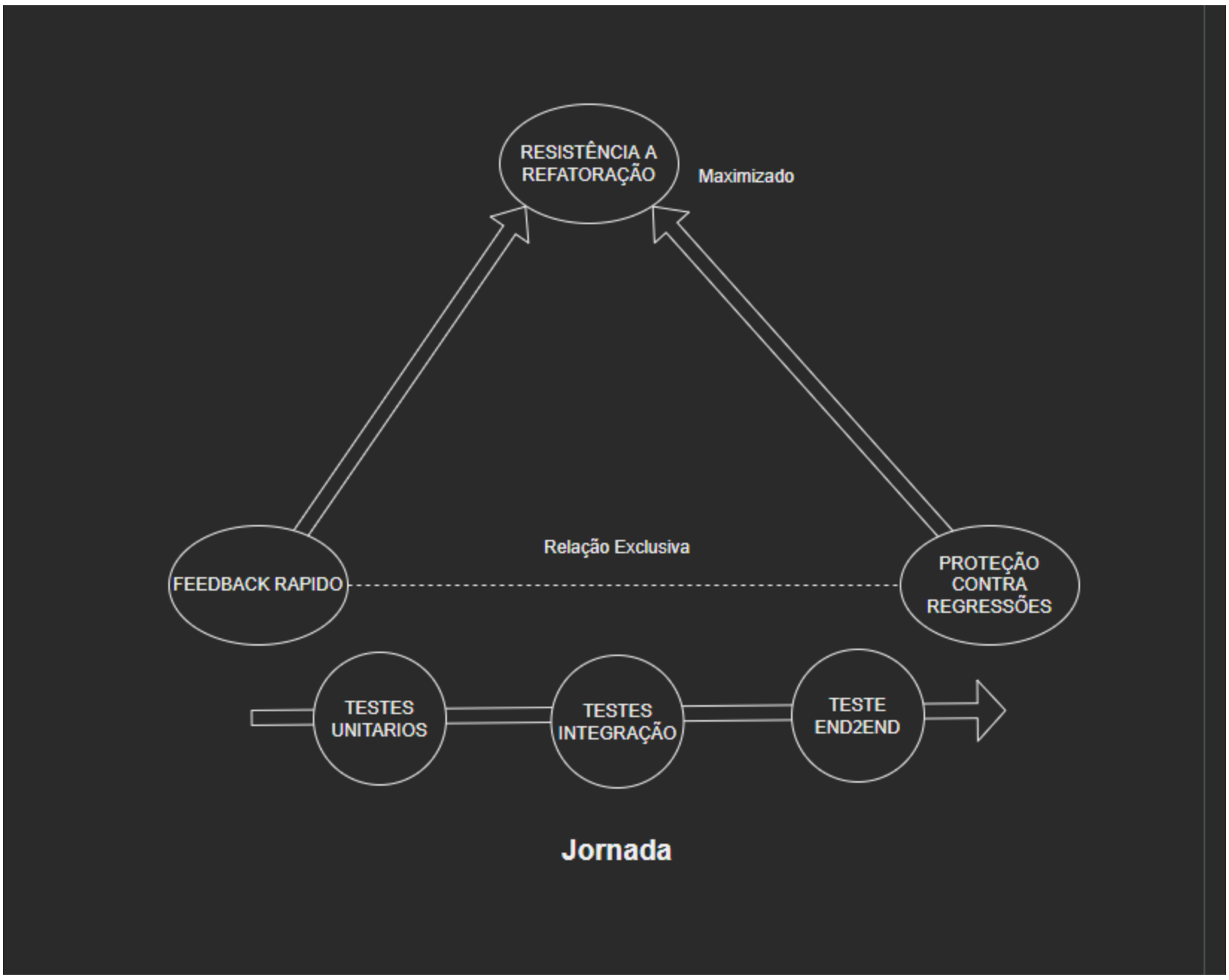
Uma das características da pirâmide de testes, e que ela e criar apartir de dois parâmetros o primeiro é a quantidade de testes realização e o segundo é a jornada do usuário.



uma pirâmide que tem três divisões, a primeira e em baixo testes unitários, do meio testes integração e no topo testes ponta a ponta, e tem uma flexa que relaciona essa ordem a jornada de testes, e a outra flexa embaixo que relaciona a quantidade de testes ao afunilamento da pirâmide conforme ela sobe

A pirâmide de testes tem uma relação, intrínseca com os pilares de testes, que conforme vamos avançando nos testes vão sendo favorecidos alguns pilares. por exemplo já foi comentado que sempre tentamos maximizar o pilar resistência contra a refatoração, é os outros pilares teríamos que decidir qual pois eles acabam sendo exclusivos entre si, no inicio da jornada temos maior favorecimento do pilar de feedback rápidos e conforme nos vamos avançando vamos perdendo,

enquanto para o proteção contra regressões temos uma relação inversa, onde temos maior favorecimento desce pilar no final da jornada do que no inicio.



Temos um triangulo formado por eclipses, onde cada um tem um nome de um pilar no topo resistência a refatoração , que tem um indicativo de esta maximizado, e tem flexa sendo apontada para eles dos outros eclipses, o pilar de feedback rápido que se encontra a esquerda e o da direita o proteção contra regressão, e a um linha tracejada que conecta também essas elipses, com o indicativo de relaçã0 exclusiva. embaixo desse triângulo, temos um flexa da esquerda para a direita e dentro da flexa a três círculos ao longo dela, o primeiro circulo, escrito testes unitários, o segundo testes de integração e o ultimo testes ponta a ponta.