

Testes de unidade

Por quê, como e quando?

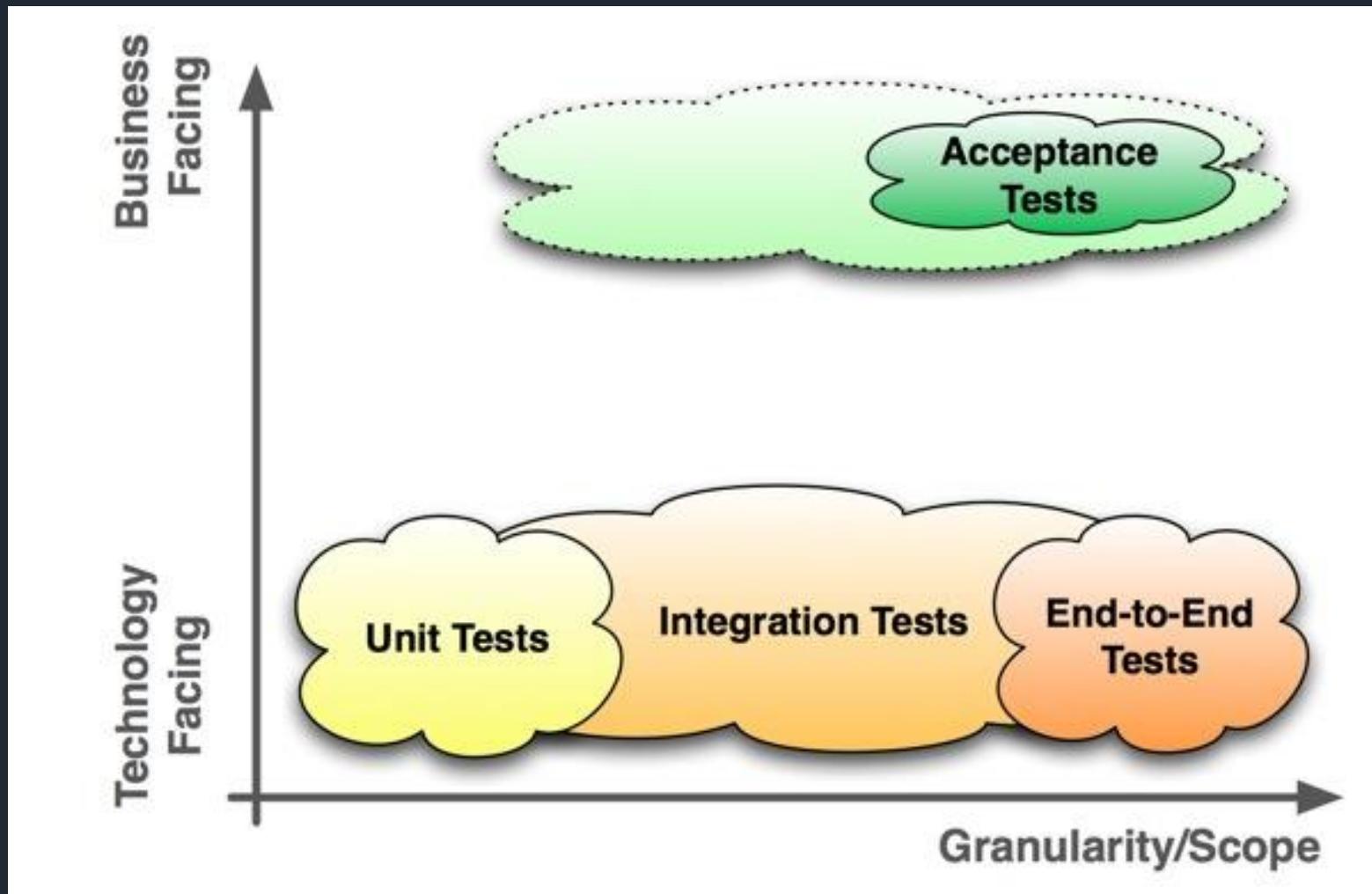
Alessandro Palmeira

Quem sou eu

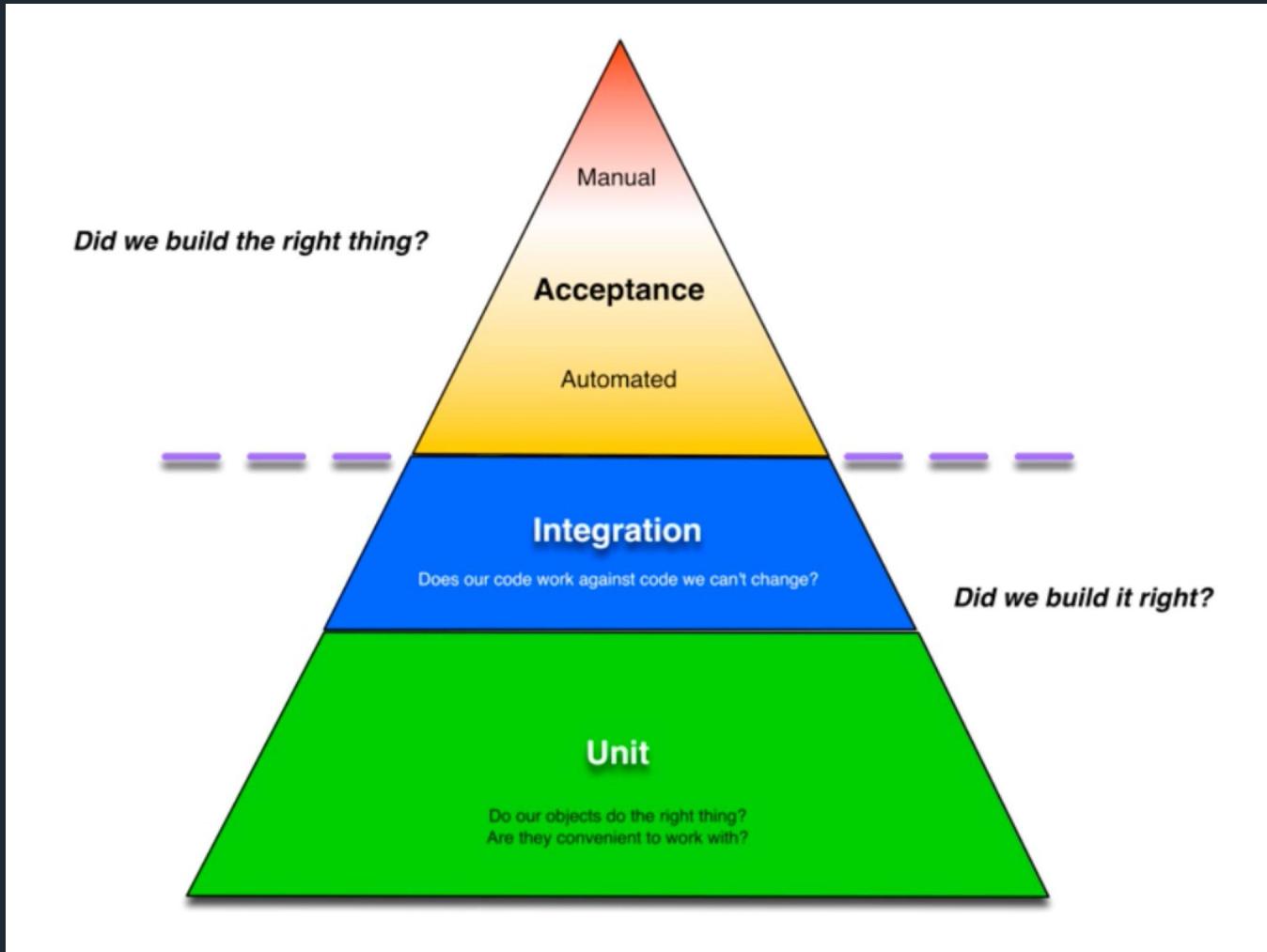
- IME-USP e Mezuro
- Caelum
- Gamers Club
- FlixBus



Escopo de testes



Escopo de testes



Por quê?

Por quê escrever testes?

- **Sério, por quê? Reflitem aí e respondam nesse link:**
<https://forms.gle/anmUyY2bN7QLiuBc8>
 - [<clique aqui para 30 segundos de música de elevador>](#)

Por quê escrever testes?

Resposta MAIS VOTADA:

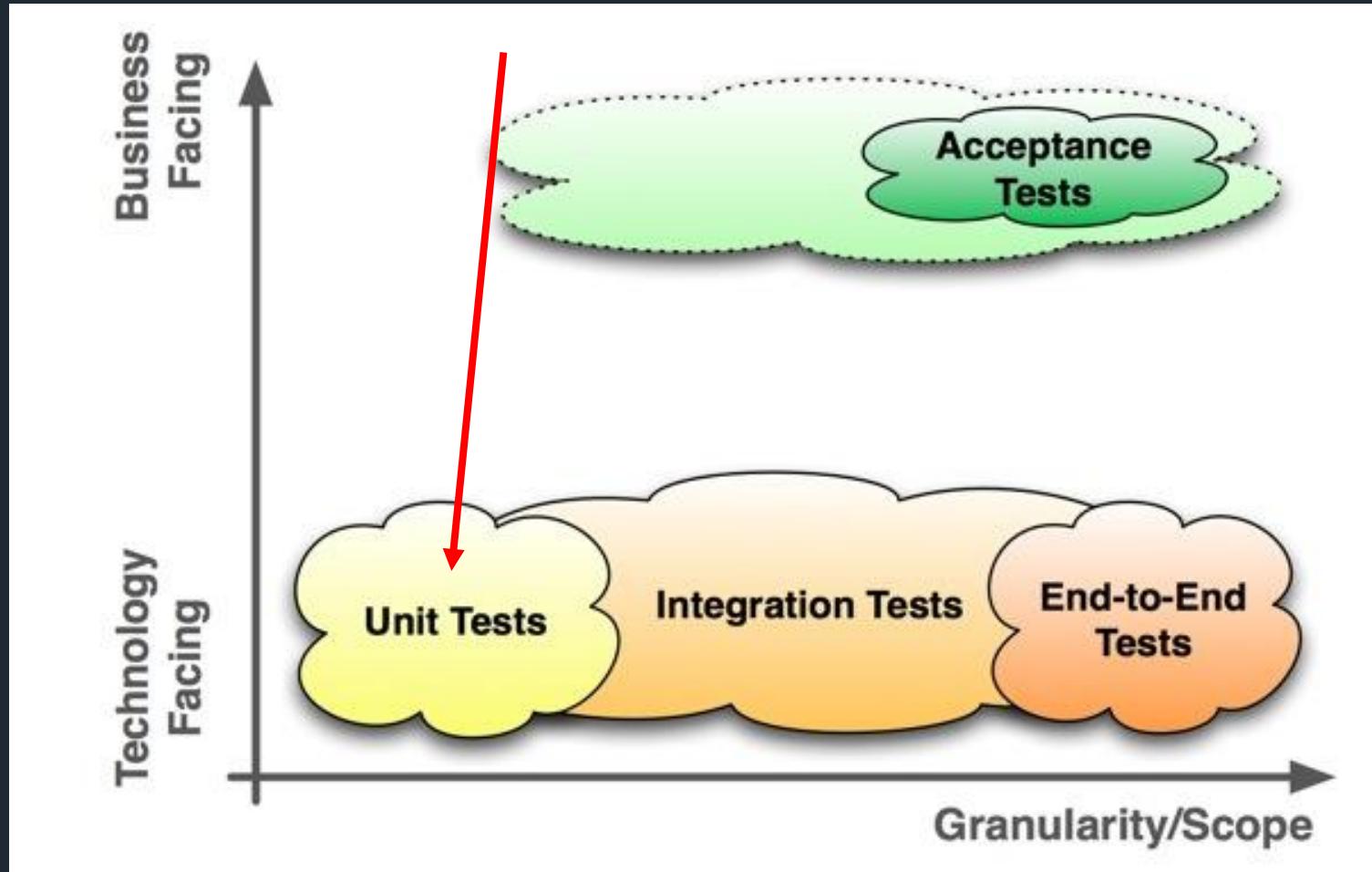
- "pq é bom ter TDD no currículo"

Por quê escrever testes?

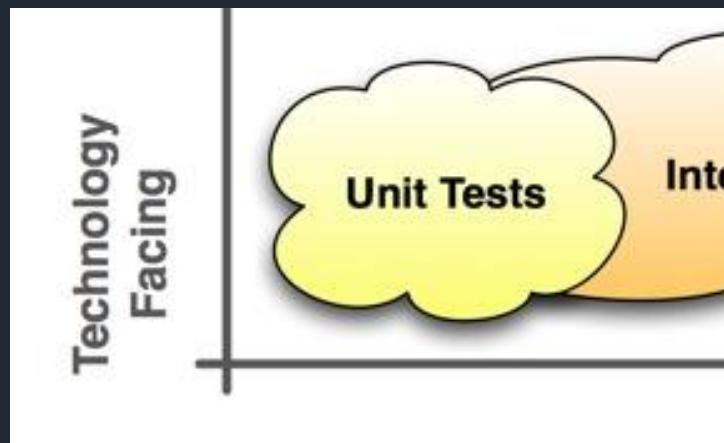
Outras respostas:

- ★ - Evitar bugs/problemas
 - "Pra eu não ter que abrir Bugs pra vocês corrigirem", QA
 - "Escrever testes te faz pensar nos possíveis problemas do seu código e prevení-los", Dev
- ★ - Garantir que o código novo não quebrou o existente
 - "Para identificar impactos indesejados em partes de código afetadas por mudanças feitas", Dev
 - "Escrever teste diminui a ansiedade (Quando você tem medo de mexer em algo porque não sabe se vai quebrar, ou subir pra prod)", Dev
 - "Garantir que qualquer alteração futura, em qualquer área do sistema, não quebre o código;", Dev
 - "Prevenir impactos negativos que futuras mudanças possam causar", Dev
- ★ - Garantir que o código novo está funcionando, Alessandro
- ★ - Aumentar a qualidade da aplicação
 - "Escrever testes aumenta a qualidade da entrega e assegura seu funcionamento", Dev
 - "Pois melhora a confiança na mudança de algum código, então caso alguém modifique o teste pode identificar que pode dar algum problema", Dev
 - "Manter a consistência da aplicação", Dev
- ★ - Testes AUTOMATIZADOS!!
 - "Pois ninguém vai testar tudo a mesma coisa todo tempo, então o teste poupa o tempo de ficar retestando oq ja teoricamente estava testado", Dev
 - "Pois somente testar manualmente ou através de prs não são sempre eficazes", Dev

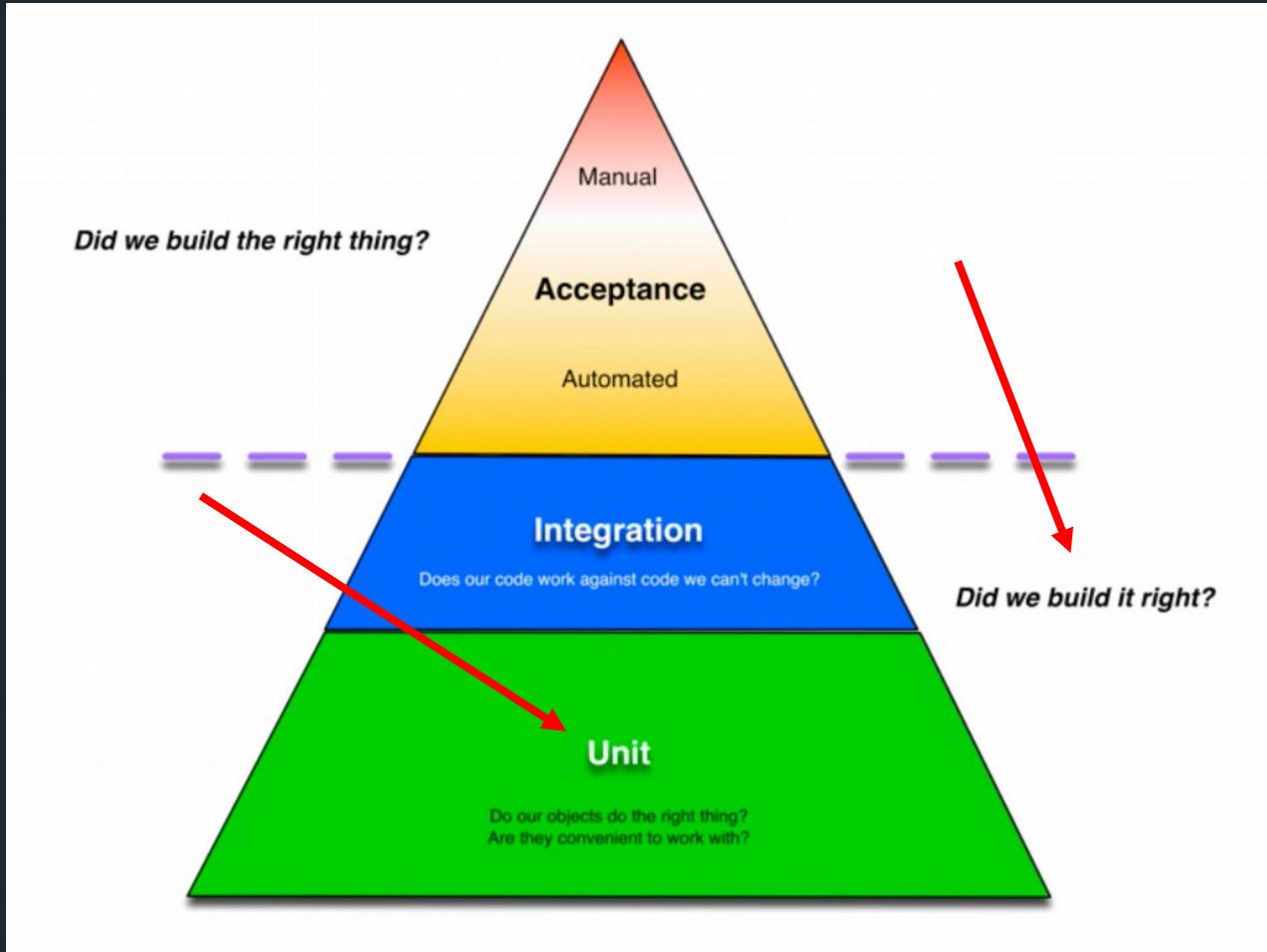
Por quê escrever testes DE UNIDADE?



Por quê escrever testes DE UNIDADE?



Por quê escrever testes DE UNIDADE?



<https://gkedge.gitbooks.io/javascript-acceptance-testing/content/>

Por quê escrever testes DE UNIDADE?

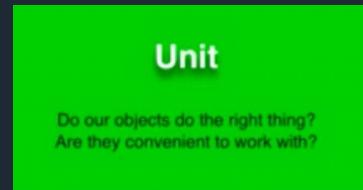
Did we build it right?

Unit

Do our objects do the right thing?
Are they convenient to work with?

Por quê escrever testes DE UNIDADE?

- Did we build it right?



Por quê escrever testes DE UNIDADE?

- **Did we build it right?**
- **Do our objects do the right thing?**

Por quê escrever testes DE UNIDADE?

- **Did we build it right?**
- **Do our objects do the right thing?**
- **Are they convenient to work with?**

EXEMPLOS

SBKing



Por quê escrever testes DE UNIDADE?

- O quê esse método faz?

`HandEvaluations::isBalanced`

Por quê escrever testes DE UNIDADE?

- O quê esse método faz?

`HandEvaluations::isBalanced`

- Uma mão com 5 espadas, 3 copas, 3 ouros e 2 paus (5,3,3,2) retorna true?
- Uma mão com 5 espadas, 4 copas, 2 ouros e 2 paus (5,4,2,2) retorna true?

Por quê escrever testes DE UNIDADE?

- **Teste de unidade como DOCUMENTAÇÃO**

Por quê escrever testes DE UNIDADE?

- **O quê essa classe faz?**

CagandoNoBequinhoScreen.java

Por quê escrever testes DE UNIDADE?

- **O quê essa classe faz?**

CagandoNoBequinhoScreen.java

- **O quê acontece se a linha 37
retornar null?**

Por quê escrever testes DE UNIDADE?

- **Uma classe melhor organizada***

Trick.java

Por quê escrever testes DE UNIDADE?

- **Teste de unidade como DOCUMENTAÇÃO**
- **Teste de unidade como AJUDA NA IDENTIFICAÇÃO DE POUCA COESÃO (e outros princípios importantes)**
- **Podemos falar de outros aspectos durante a discussão**

COMO

Como escrever testes DE UNIDADE?

Test Desiderata



Kent Beck [Follow](#)

Oct 18, 2019 · 2 min read

[!\[\]\(2fff3c9e4afb192f04e878e77fb552ed_img.jpg\)](#) [!\[\]\(424424d14aba49c86d1c60a64247128f_img.jpg\)](#) [!\[\]\(c6faaa65325805483013e4324c1f0b1d_img.jpg\)](#)

https://medium.com/@kentbeck_7670/test-desiderata-94150638a4b3

Como escrever testes DE UNIDADE?

- Isolated** – tests should return the same results regardless of the order in which they are run.
- Composable** – if tests are isolated, then I can run 1 or 10 or 100 or 1,000,000 and get the same results.
- Fast** – tests should run quickly.
- Inspiring** – passing the tests should inspire confidence
- Writable** – tests should be cheap to write relative to the cost of the code being tested.
- Readable** – tests should be comprehensible for reader, invoking the motivation for writing this particular test.
- Behavioral** – tests should be sensitive to changes in the behavior of the code under test. If the behavior changes, the test result should change.
- Structure-insensitive** – tests should not change their result if the structure of the code changes.
- Automated** – tests should run without human intervention.
- Specific** – if a test fails, the cause of the failure should be obvious.
- Deterministic** – if nothing changes, the test result shouldn't change.
- Predictive** – if the tests all pass, then the code under test should be suitable for production

COMO (em
Javascript/Typescript)

Como escrever testes DE UNIDADE (js) ?

The screenshot shows the official Jest website. At the top left is a green button labeled "JEST 24.9". To the right are links for "Docs", "API", "Help", "Blog", and "English", along with a search bar and a GitHub link. A star icon indicates 29,130 stars. Below the header is a graphic featuring three cards: one with a red "FAIL" icon, one with a green "PASS" icon, and one with a green checkmark icon. Buttons for "GET STARTED", "DOCS", "CONFIG", and "GET HELP" are visible. The main text area below the graphic reads: "Jest is a delightful JavaScript Testing Framework with a focus on simplicity. It works with projects using: [Babel](#), [TypeScript](#), [Node](#), [React](#), [Angular](#), [Vue](#) and more!"

JEST 24.9

Docs API Help Blog English GitHub

★ Star 29,130

FAIL PASS JEST

GET STARTED DOCS CONFIG GET HELP

Jest is a delightful JavaScript Testing Framework with a focus on simplicity.

It works with projects using: [Babel](#), [TypeScript](#), [Node](#), [React](#), [Angular](#), [Vue](#) and more!

Como escrever testes DE UNIDADE (js) ?

SINON.JS

Documentation Releases Guides How To 



Standalone test spies, stubs and mocks for JavaScript.
Works with any unit testing framework.

[GET STARTED](#)

 Star Sinon.JS on Github

Como escrever testes DE UNIDADE (js) ?

faker.js - generate massive amounts of fake data in the browser and node.js

faker.js

Como escrever testes DE UNIDADE (js) ?

```
type IDataMock<T> = {
  [P in keyof T] ?: IDataMock<T[P]>
}

const dataMock = <T>(instance: IDataMock<T>): T => {
  return instance as any
}

export { dataMock }
```

```
const accessTokenMock = dataMock<AccessTokenStructure>({
  id: faker.random.number(),
  profiles: []
})
```

Como escrever testes DE UNIDADE (js) ?

```
const stub = <T extends any> (): T => {
  const typeAssertion = <T>{}

  for (const prop in typeAssertion) {
    if (typeAssertion.hasOwnProperty(prop)) {
      typeAssertion[prop] = undefined
    }
  }

  return typeAssertion
}

export { stub }
```

```
let teamRepository: TeamRepository
let accountRepository: AccountRepository
let teamMemberRepository: TeamMemberRepository

beforeEach(() => {
  teamRepository = stub<TeamRepository>()
  accountRepository = stub<AccountRepository>()
  teamMemberRepository = stub<TeamMemberRepository>()
})
```

QUANDO?

Referências

- Maurício Aniche: “**Unidade, integração ou sistema? Qual teste fazer?**” [link](#)
- Lucas Santos: “**Testes Unitários 101: Mocks, Stubs, Spies e todas essas palavras difíceis**” [link](#)
- Kent Beck: “**Test Desiderata**” (inglês) [link](#)

Obrigado!