HUGE

Hello.

October 13, 2016
Web.br
hugeinc.com

13 de outubro, 2016

Desenvolvendo aplicações de qualidade com TDD

Huge

Isabella Silveira

Web Engineer @ Huge.

Front-End, Back-End, Agile
e Integração Contínua.

HUGE


WAY
ONE


AMERICAN
EXPRESS


COMCAST


Google


GUCCI


LOWE'S


DIAGEO


VANS


TED


Unilever


TOYOTA


FOX


GE


Morgan Stanley


LVMH


Pfizer


THOMSON REUTERS


SAMSUNG


Coca-Cola


hulu


Huge
Brews


Hello.


Fuckin' eh.


~~Get rich.~~
Die trying.
HUGE


My other bag
is your mom.
HUGE

Agenda.

↓

1. Introdução
2. O que é TDD? 3. Por que usar TDD?
4. Show me the code.
5. Boas práticas. 6. Ferramentas.
7. Próximos passos.
8. Conclusão.

Introdução.

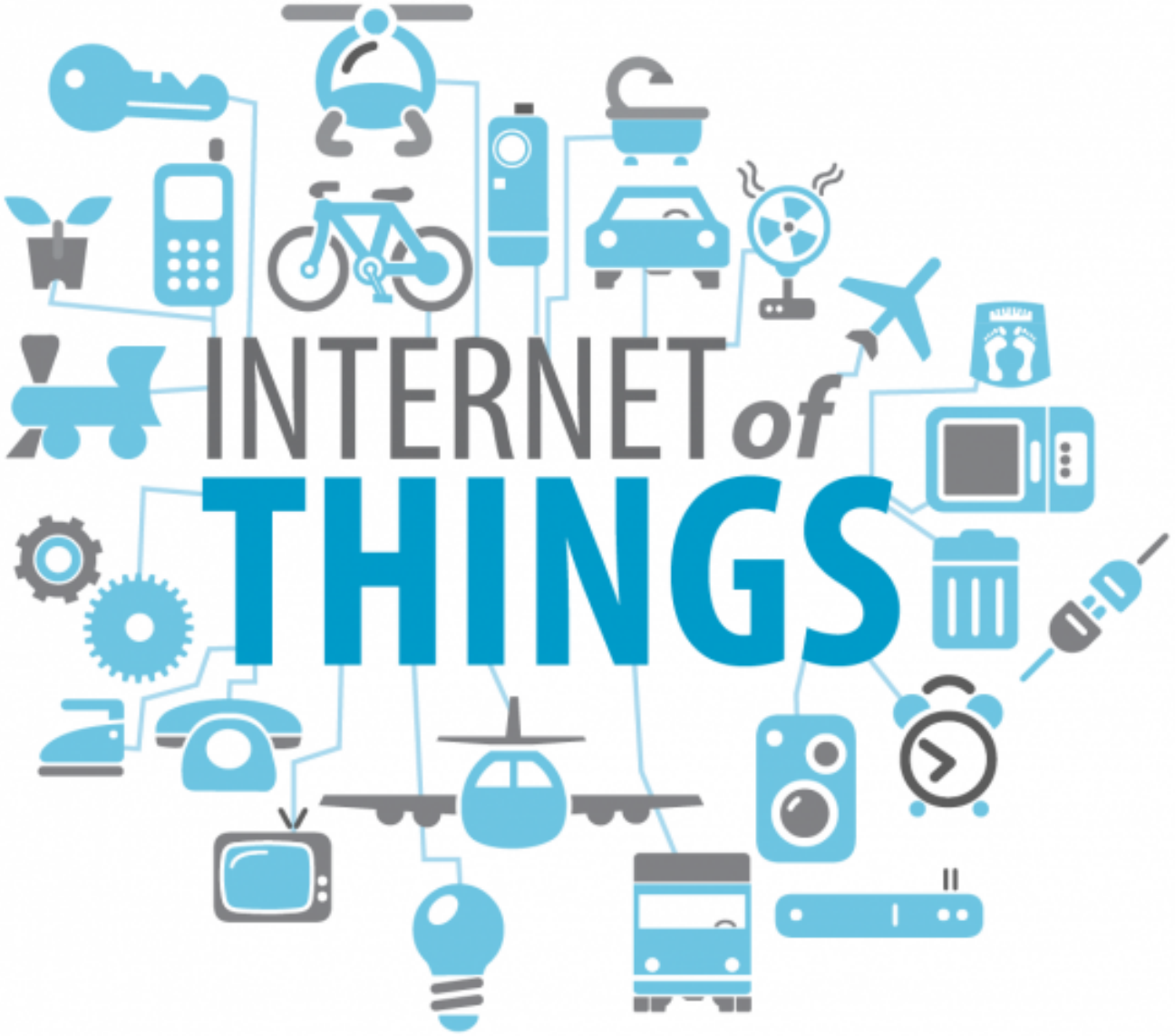
Hoje em dia, a qualidade é um diferencial em tudo que consumimos.

E podemos dizer que o mundo moderno funciona a base de código.


A relax
15:46
MODE
MODE
AUTO
MAX


iRobot
Roomba
CLEAN
DOCK
HOUR
MINUTE
DAY


NASA
NORTHROP GRUMMAN
871
DRYDEN FLIGHT RESEARCH CENTER


INTERNET of
THINGS

Qualidade de software é uma preocupação crescente.

TDD é uma ótima ferramenta para nos ajudar a alcançar este objetivo.

O que é TDD?

Vamos falar de coisa boa.

Testes de Software.

Tipos de teste de software.

Tipos de teste de software:

1. Testes unitários.

2. Testes de integração.

3. Testes de aceitação.

Tipos de teste de software:

1. Testes unitários.
2. Testes de integração.
3. Testes de aceitação.

Garantem o funcionamento de cada parte individual do sistema.

Tipos de teste de software.

1. Testes unitários.

2. Testes de integração.

3. Testes de aceitação.

Asseguram que duas ou mais unidades funcionam corretamente quando trabalhando juntas.

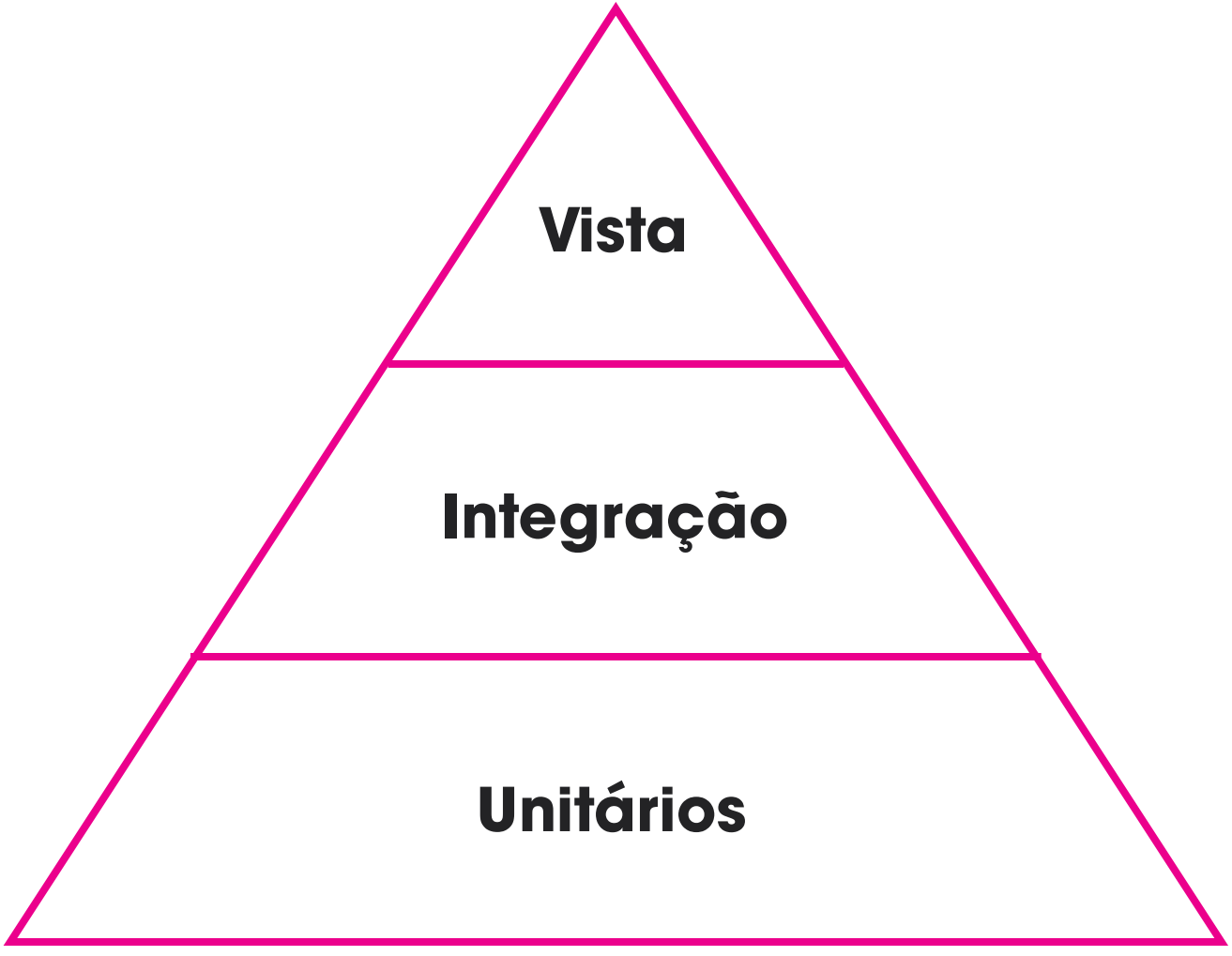
Tipos de teste de software.

1. Testes unitários.

2. Testes de integração.

3. Testes de aceitação.

Testes na vista, simulam o comportamento do usuário.


Vista
Integração
Unitários

O que é TDD?

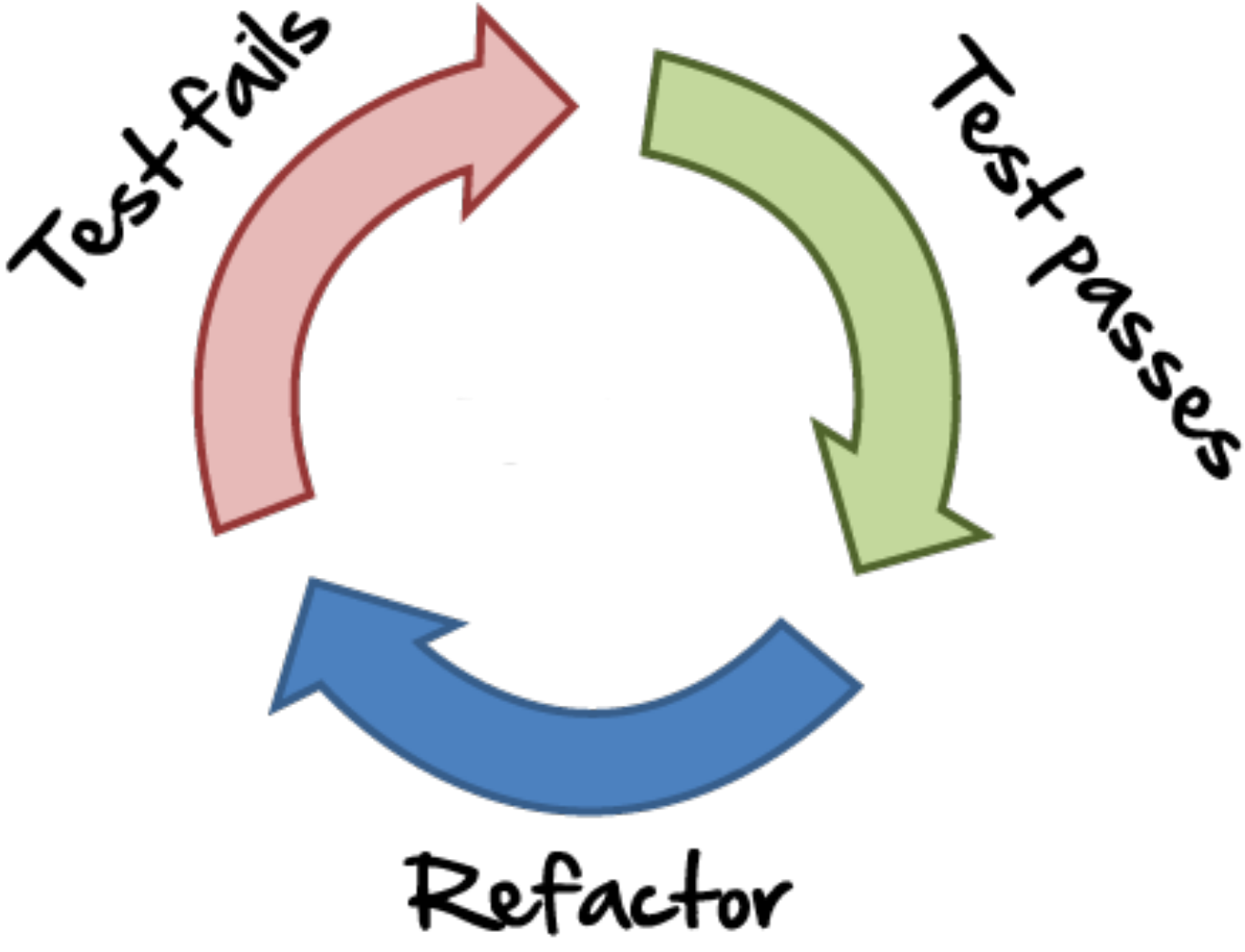
Metodologia de desenvolvimento de software onde os testes são escritos antes das funcionalidades em si.

O que é TDD?

As três leis do TDD.

As três leis do TDD:

Você deve escrever um teste que falhe antes de escrever qualquer código de produção.


Test fails
Test passes
Refactor

"Getting software to work is only half of the job."

Robert C. Martin (Uncle Bob)

Primeiro fazer o código funcionar, depois melhorar a estrutura.

As três leis do TDD:

Você não deve escrever um teste que contenha mais do que o necessário para simular o comportamento da função.

Keep it simple: faça apenas o necessário no escopo da função.

As três leis do TDD:

Você não deve escrever mais código de produção do que o suficiente para fazer o teste passar.

“A failing test should read like a high-quality bug report.“

Eric Elliott

“ Mas como eu sei o que testar?”

Divida a sua user story em features, e as features em unidades atômicas.

“Eu, enquanto usuário, gostaria de ser redirecionado para a home depois de logar no sistema.”

Definição de feature:

Form de login.

Próximo passo:

Definir as funcionalidades e o comportamento esperado dessa feature.

```
describe('LoginForm', function() {

    it('should identify invalid email addresses', function() {});
    it('should validate valid email addresses', function() {});
    it('should change path when submitting a form', function() { });

    it('should submit form when hitting enter on the password field', function() { });

});
```

Por que usar TDD?

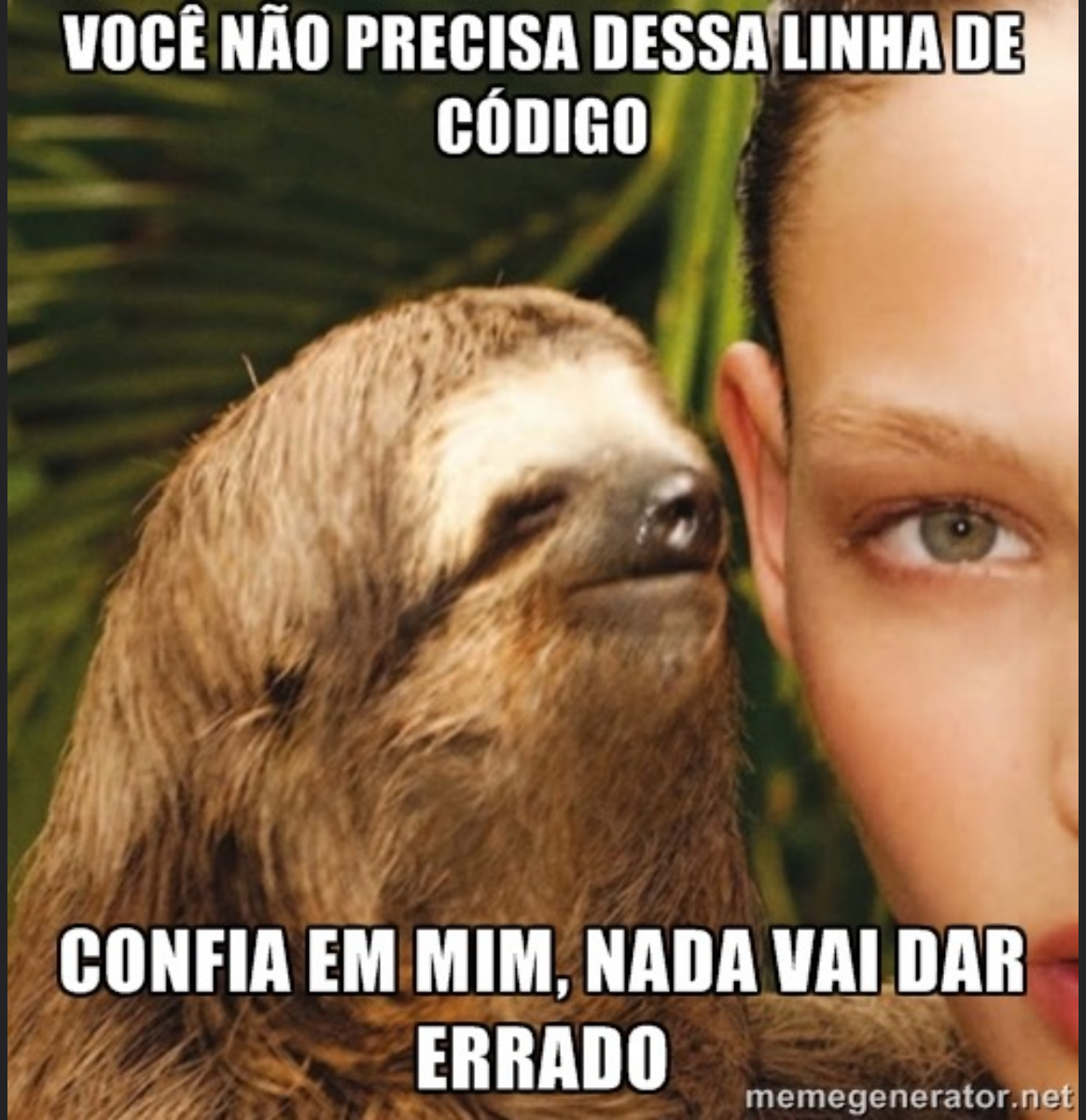
Por que usar TDD?

1. Auxilia no design da API.
2. Testes = Documentação.
3. Código de qualidade.
4. Viabiliza IC.

Por que usar TDD?

1. Auxilia no design da API.
2. Testes = Documentação.
3. Código de qualidade.
4. Viabiliza IC.

TDD te dá uma perspectiva mais clara do que realmente é necessário.


VOCÊ NÃO PRECISA DESSA LINHA DE CÓDIGO
CONFIA EM MIM, NADA VAI DAR ERRADO
memegenerator.net

TDD reduz a complexidade do código.

Por que usar TDD?

1. Auxilia no design da API.
2. Testes = Documentação.
3. Código de qualidade.
4. Viabiliza IC.

Cada teste documenta com exatidão o comportamento de cada função.

Por que usar TDD?

1. Auxilia no design da API.
2. Testes = Documentação.
3. Código de qualidade.
4. Viabiliza IC.

Cada linha de código testado é uma linha de código confiável.

Por que usar TDD?

1. Auxilia no design da API.
2. Testes = Documentação.
3. Código de qualidade.
4. Viabiliza IC.

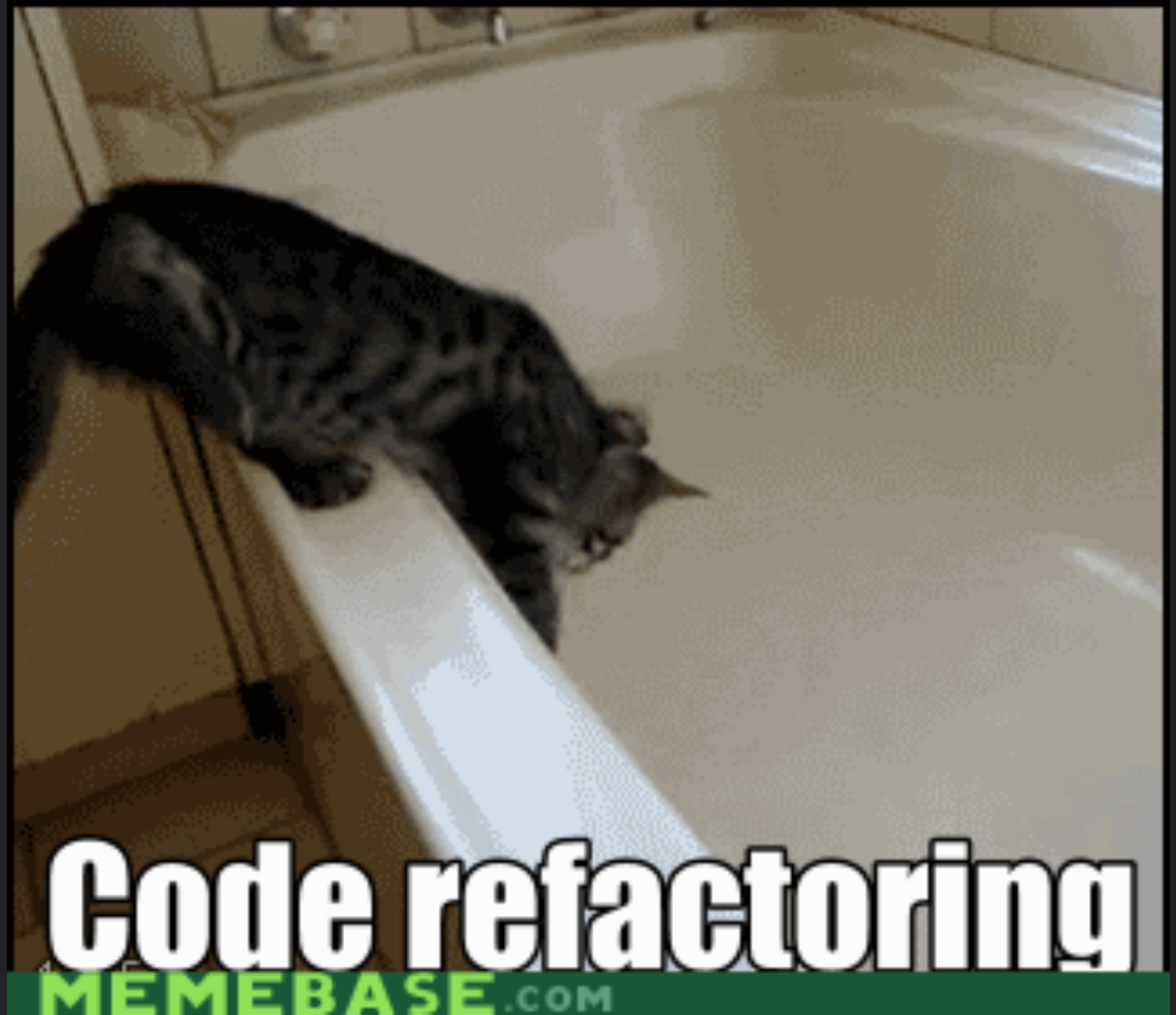
A integração contínua fornece o panorama geral da aplicação.


Backup-Fauno #115
Gnosys-Historico #532
139 tests run, 0 test failed
Gnosys-Restaurante #601
84 tests run, 0 test failed
Deploy-HOMO #452
Gnosys-Indicadores #525
7 tests run, 0 test failed
Gnosys-Seguranca #794
57 tests run, 0 test failed
Gnosys-Avaliacao #809
182 tests run, 0 test failed
Gnosys-Inscricao #815
408 tests run, 6 tests failed
Gnosys-SGA #1014
460 tests run, 0 test failed
Gnosys-Bolsa #1318
824 tests run, 19 tests failed
Gnosys-node-gerencia #25
18 tests run, 0 test failed
Gnosys-Turma #290
11 tests run, 0 test failed
Gnosys-CadastroGenerico-Artifac
Gnosys-ParserWiki #398
19 tests run, 0 test failed
Monitoramento #21600
18 tests run, 4 tests failed
Gnosys-CadastroGenerico-Matrix
Gnosys-Pessoa #820
177 tests run, 0 test failed
Puppet #259
Gnosys-Commons #1246
378 tests run, 0 test failed
Gnosys-Portal #943
89 tests run, 0 test failed
SIGA #1445
2543 tests run, 0 test failed
Gnosys-Comunicacao #809
99 tests run, 4 tests failed
Gnosys-PreMatricula #1511
374 tests run, 0 test failed
SQL-Job-Runner #1625
Gnosys-Documentos #637
138 tests run, 0 test failed
Gnosys-Questionario #389
33 tests run, 0 test failed
SQL-Metadata #90
Gnosys-Ensino #456
27 tests run, 0 test failed
Gnosys-Registro #829
317 tests run, 3 tests failed
SAMSUNG

Testes previnem código quebrado.

Testes mapeiam cada parte do comportamento da sua aplicação.

Testes acabam com o medo de refactorings.


Code refactoring
MEMEBASE.COM

TDD facilita a identificação de bugs de regressão.

TDD estimula designs modulares.

Show me the code.

Quatro perguntas primordiais.

Quatro perguntas primordiais.

1. O que eu estou testando?

2. Quais os parâmetros?

3. Qual a saída obtida?

4. Qual a saída esperada?

Exemplo prático: logIntoSystem()

Red.

```
describe('Login System', function() {
    describe('logIntoSystem', function() {
        it('Should return "Hey" + username if successfully logged into the system', function() {

            //setup inicial do teste
            var users = [
                    {
                      "username":"jude",
                      "password": "naaaaaaNaaaNaaaNaNaNaNa"
                    }
                ],
                username = "jude",
                password = "naaaaaaNaaaNaaaNaNaNaNa",
                resultadoEncontrado,
                resultadoEsperado = "Hey jude";

            //chamando a função real
            resultadoEncontrado = LoginModule.logIntoSystem(users, username, password);

            //fazendo a asserção
            expect(resultadoEncontrado).to.equal(resultadoEsperado);
        });
    });
});
```

```
describe('Login System', function() {
    describe('logIntoSystem', function() {
        it('Should return "Hey" + username if successfully logged into the system', function() {

            //setup inicial do teste
            var users = [
                    {
                      "username":"jude",
                      "password": "naaaaaaNaaaNaaaNaNaNaNa"
                    }
                ],
                username = "jude",
                password = "naaaaaaNaaaNaaaNaNaNaNa",
                resultadoEncontrado,
                resultadoEsperado = "Hey jude";

            //chamando a função real
            resultadoEncontrado = LoginModule.logIntoSystem(users, username, password);

            //fazendo a asserção
            expect(resultadoEncontrado).to.equal(resultadoEsperado);
        });
    });
});
```

```
describe('Login System', function() {
    describe('logIntoSystem', function() {
        it('Should return "Hey" + username if successfully logged into the system', function() {

            //setup inicial do teste
            var users = [
                    {
                      "username":"jude",
                      "password": "naaaaaaNaaaNaaaNaNaNaNa"
                    }
                ],
                username = "jude",
                password = "naaaaaaNaaaNaaaNaNaNaNa",
                resultadoEncontrado,
                resultadoEsperado = "Hey jude";

            //chamando a função real
            resultadoEncontrado = LoginModule.logIntoSystem(users, username, password);

            //fazendo a asserção
            expect(resultadoEncontrado).to.equal(resultadoEsperado);
        });
    });
});
```

```
1) Should return "Hey" + username if successfully logged into the system
     Login System logIntoSystem
     TypeError: 'undefined' is not a function (evaluating 'LoginModule.logIntoSystem(users, username, password)')
    at http://localhost:9876/context.html:21
```

Green.

```
function logIntoSystem(users, username, password) {
    return "Hey jude";
}
```

```
Login System
  logIntoSystem
    ✓ Should return "Hey" + username if successfully logged into the system
```

Refactor.

```
function logIntoSystem(users, username, password) {
    for (var i = 0; i < users.length; i++) {
        if (users[i].username == username) {
            if (users[i].password == password) {
                return "Hey" + users[i].username;
            }
        }
    }
}
```

```
Login System
  logIntoSystem
    ✓ Should return "Hey" + username if successfully logged into the system
```

Testar o comportamento em caso de falha também é importante.

```
describe('Login System', function() {
    describe('logIntoSystem', function() {
        it('Should return "Hey" + username + ", refrain" if the password is wrong', function() {

            //setup inicial do teste
            var users = [
                    {
                      "username":"jude",
                      "password": "naaaaaaNaaaNaaaNaNaNaNa"
                    }
                ],
                username = "jude",
                password = "senhaErrada",
                resultadoEncontrado,
                resultadoEsperado = "Hey jude, refrain";

            //chamando a função real
            resultadoEncontrado = LoginModule.logIntoSystem(users, username, password);

            //fazendo a asserção
            expect(resultadoEncontrado).to.equal(resultadoEsperado);
        });
    });
});
```

```
describe('Login System', function() {
    describe('logIntoSystem', function() {
        it('Should return "Hey" + username + ", refrain" if the password is wrong', function() {

            //setup inicial do teste
            var users = [
                    {
                      "username":"jude",
                      "password": "naaaaaaNaaaNaaaNaNaNaNa"
                    }
                ],
                username = "jude",
                password = "senhaErrada",
                resultadoEncontrado,
                resultadoEsperado = "Hey jude, refrain";

            //chamando a função real
            resultadoEncontrado = LoginModule.logIntoSystem(users, username, password);

            //fazendo a asserção
            expect(resultadoEncontrado).to.equal(resultadoEsperado);
        });
    });
});
```

```
describe('Login System', function() {
    describe('logIntoSystem', function() {
        it('Should return "Hey" + username + ", refrain" if the password is wrong', function() {

            //setup inicial do teste
            var users = [
                    {
                      "username":"jude",
                      "password": "naaaaaaNaaaNaaaNaNaNaNa"
                    }
                ],
                username = "jude",
                password = "senhaErrada",
                resultadoEncontrado,
                resultadoEsperado = "Hey jude, refrain";

            //chamando a função real
            resultadoEncontrado = LoginModule.logIntoSystem(users, username, password);

            //fazendo a asserção
            expect(resultadoEncontrado).to.equal(resultadoEsperado);
        });
    });
});
```

```
function logIntoSystem(users, username, password) {
    for (var i = 0; i < users.length; i++) {
        if (users[i].username == username) {
            if (users[i].password == password) {
                return "Hey" + users[i].username;
            }
            else {
                return "Hey" + users[i].username + ", refrain";
            }
        }
    }
}
```

Boas práticas.

Cada teste deve ser independente.

Não faça asserções desnecessárias.

Mocke os estados e recursos externos.

Use nomes e descrições consistentes.

Teste um comportamento de cada vez.

Ferramentas.

Frameworks de teste.

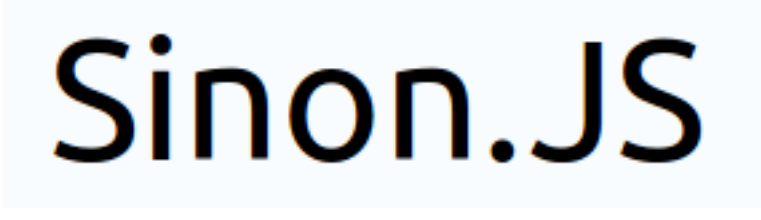

mocha


Jasmine


chai


QUnit
js unit testing


Protractor
end to end testing for AngularJS


Sinon.JS

Test runners.

KARMA

Próximos Passos.

Code Coverage.

Karma Coverage.

Integração Contínua.

CODESHIP

circleci

Bamboo

Solano Labs

snap

semaphore

AppVeyor

drone.io

Conclusão.

Qualidade agrega valor à sua aplicação.

TDD não é a única forma de garantir a qualidade do software, apesar de ser um bom aliado.

Sempre prezar por um código modularizado e testável.

E o mais importante de tudo:

Qualidade de software não é opcional.

Perguntas?

Get in touch.


isilveira@hugeinc.com


@silveira_bells


/in/isabellasilveira


/bella-silveira

HUGE

Done.