

WORKSHOP

Produzindo Especificações Ágeis e Testes Funcionais com Concordia

THIAGO DELGADO PINTO

SEMANA DE EXTENSÃO 2019

BSI - CEFET/RJ NOVA FRIBURGO

13/05/2019



Licença Creative Commons 4

Concordia

Concordia é a [deusa romana](#) que personifica a "*concordância*", o "*acordo*"¹.

a ideia que a linguagem seja usada como forma de **aproximar** clientes e desenvolvedores

1. <https://www.britannica.com/topic/Concordia-Roman-goddess>
2. [https://it.wikipedia.org/wiki/Tempio_della_Concordia_\(Agrigento\)](https://it.wikipedia.org/wiki/Tempio_della_Concordia_(Agrigento))



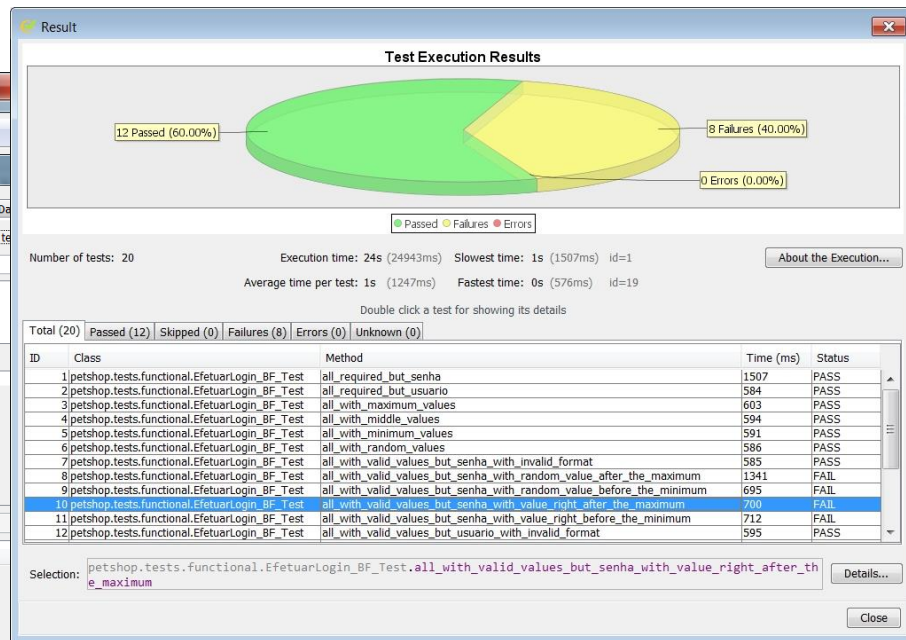
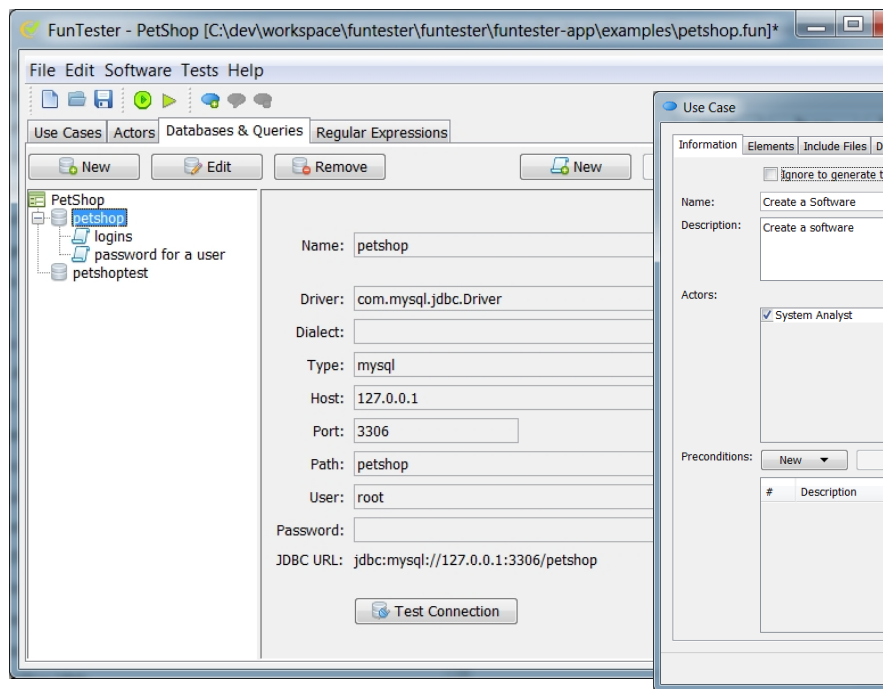
Templo de Concordia - Sicília, Itália - 440-430 a.C.²

logotipo?

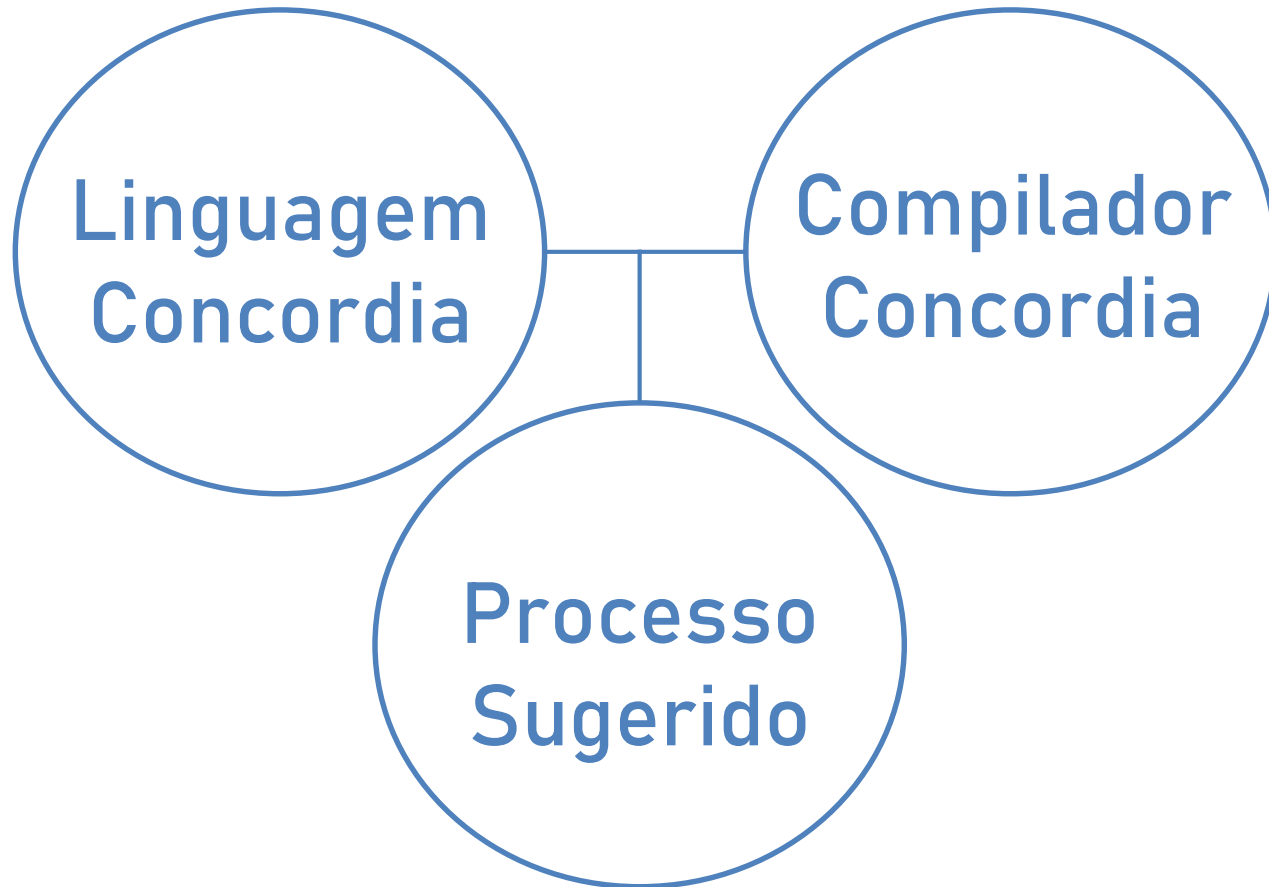




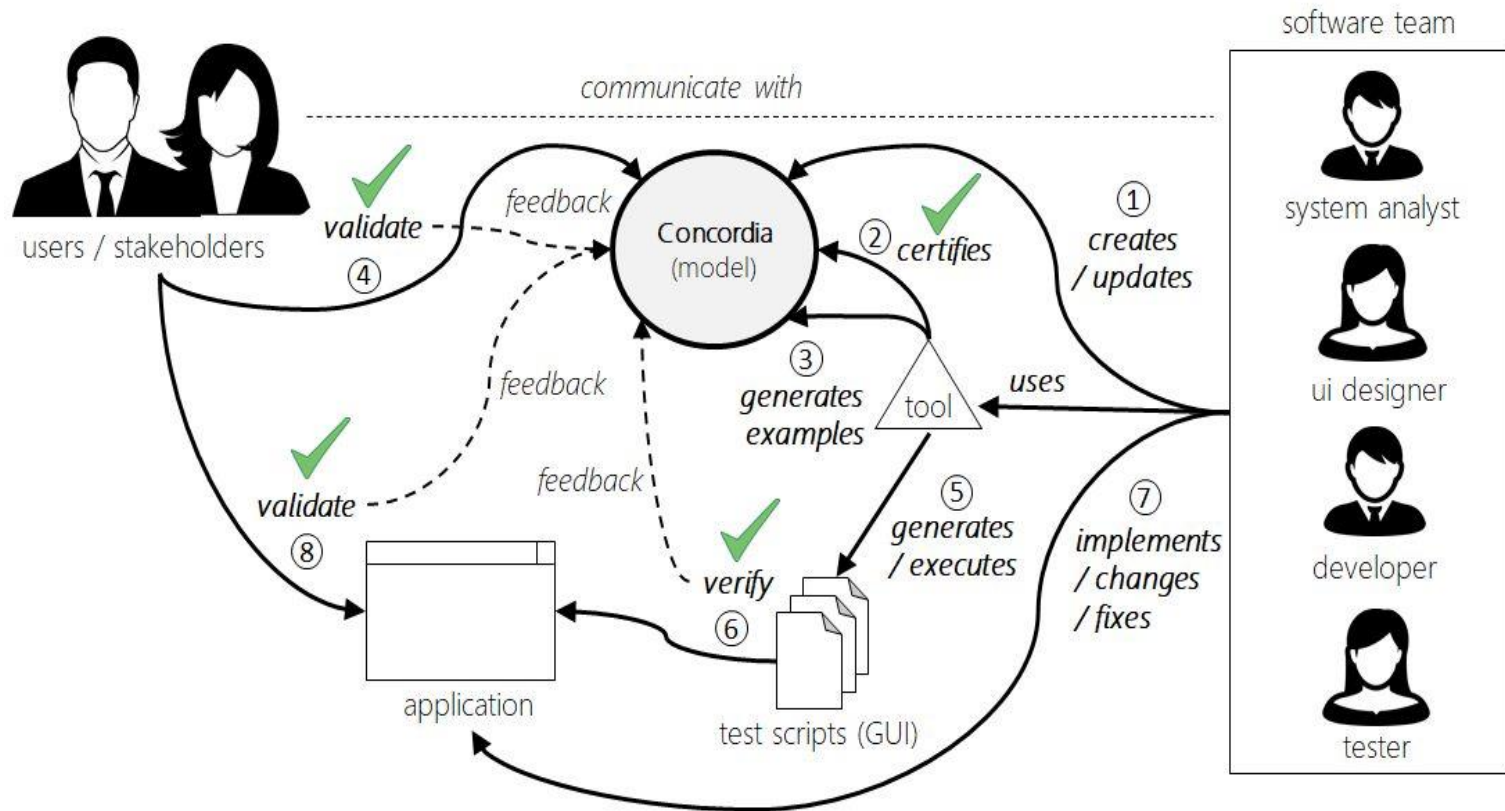
- funtester.org ou github.com/funtester
- gera testes funcionais completos para aplicações Web, Mobile e Java Swing
- opensource – escrita em Java
- desenvolvi durante meu mestrado



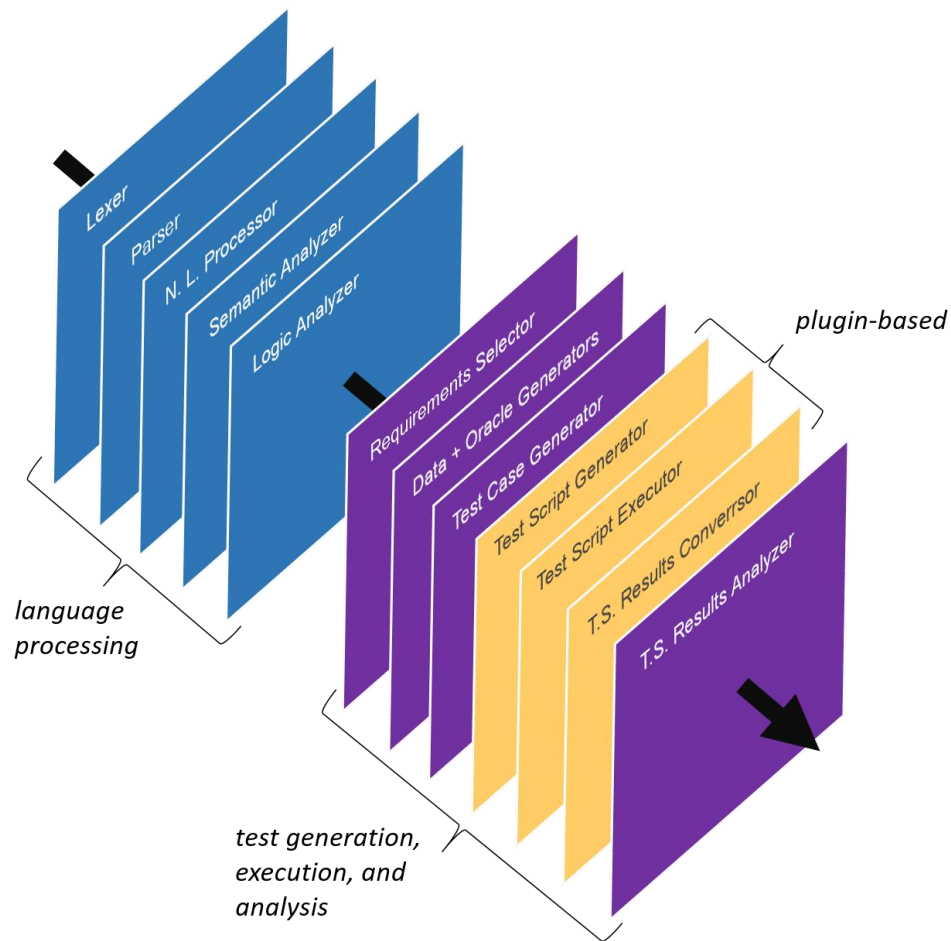
COMPOSIÇÃO

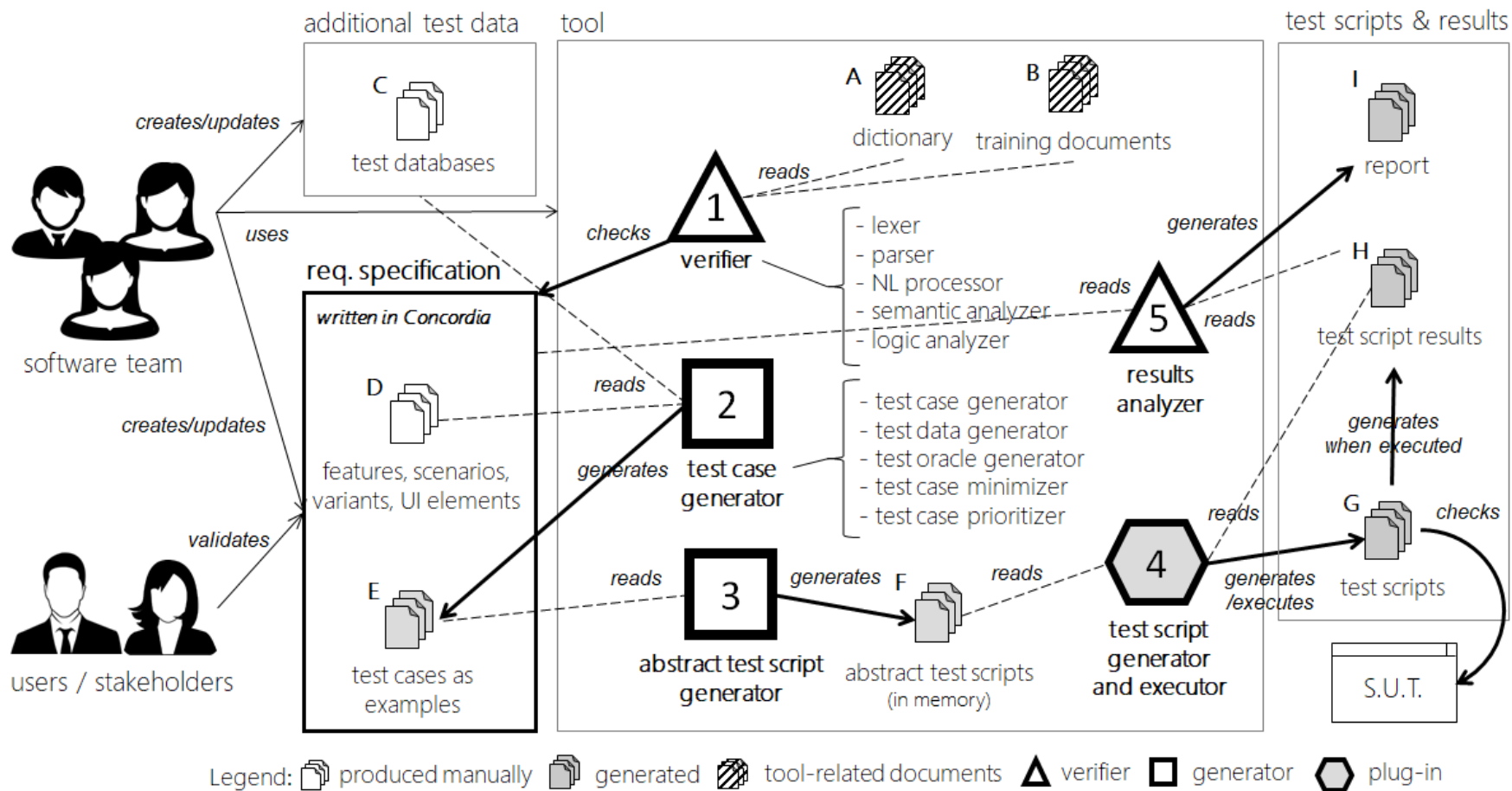


PROCESSO



ESTRUTURA







concordialang.org

DESENVOLVIMENTO



SEMANTIC
VERSIONING



v8+

Breaking
change

Bug
fix

1.2.3

New
feature

license AGPL-3.0

slack chat

```
PASS __tests__\lexer\DatabaseLexerTest.spec.ts
PASS __tests__\util\ReferenceReplacerTest.spec.ts
PASS __tests__\db\QueryParserTest.spec.ts
PASS __tests__\lexer\ImportLexerTest.spec.ts
PASS __tests__\testdata\raw\DoubleGeneratorTest.spec.ts
PASS __tests__\parser\TestCaseParserTest.spec.ts
PASS __tests__\lexer\RegexBlockLexerTest.spec.ts
PASS __tests__\lexer\StepWhenLexerTest.spec.ts
PASS __tests__\parser\TagCollectorTest.spec.ts
PASS __tests__\req\LineCheckerTest.spec.ts
PASS __tests__\testdata\random\RandomStringTest.spec.ts
PASS __tests__\testdata\random\RandomDateTest.spec.ts
PASS __tests__\testdata\random\RandomDoubleTest.spec.ts
PASS __tests__\parser\ScenarioParserTest.spec.ts
PASS __tests__\testdata\random\RandomLongTest.spec.ts
PASS __tests__\testdata\random\RandomTimeTest.spec.ts
PASS __tests__\lexer\ConstantBlockLexerTest.spec.ts
PASS __tests__\lexer\TextLexerTest.spec.ts
PASS __tests__\lexer\RegexLexerTest.spec.ts
PASS __tests__\nlp\RuleBuilderTest.spec.ts
PASS __tests__\req\ExpressionsTest.spec.ts
```

Test Suites: 81 passed, 81 total

Tests: 695 passed, 695 total

Snapshots: 0 total

Time: 16.929s, estimated 29s

Ran all test suites.

características da ferramenta

funciona como um **compilador**

usa **processamento de ling. nat.** e **aprendizagem supervisionada**
reconhecimento de intenção

gera **casos de teste**

gera e executa **scripts de teste funcional**

analisa resultados da execução

divulgando a ferramenta

mais pessoas conhecendo, mais feedback, melhor ela ficará!

usando a ferramenta

informe o que você achou, quais dúvidas que teve ou tem
mostre casos em que ela não funcionou como esperado
mostre como ela pode melhorar para ajudar você ou sua empresa

melhorando a documentação

estendendo, traduzindo, corrigindo
criando um *logotipo*!

registrando **defeitos** ou **sugestões**

criando **novos testes**

pode ser simplesmente criando pequenas variações de coisas que existem

criando **novos plug-ins**

para sua linguagem e framework de teste preferidos

colocando a mão na massa

corrigindo bugs no código

propondo alterações ou melhorias



linguagem Concordia

visão geral

características da linguagem

inspirada em [Gherkin](#)

legível para pessoas **envolvidas no negócio** (*business-readable*)

separa declarações de *negócio* das "*tecnológicas*"

baseada em **dicionário** → traduzível, expansível

permite especificar **requisitos** de qualquer tipo

permite especificar **casos de teste** e **regras de negócio**

Termo	Finalidade	Global
Funcionalidade	Descreve uma funcionalidade do ponto de vista de negócio	<i>Sim</i>
Cenário	Descreve um cenário do ponto de vista de negócio	Não
Variante	Descreve interação entre usuário e sistema em alto nível	Não
Caso de Teste	Descreve interação entre usuário e sistema em nível mais baixo	Não
Elemento de IU	Descreve um elemento da Interface de Usuário	Não*
Constantes	Descreve um bloco de constantes	<i>Sim</i>
Tabela	Descreve uma tabela	<i>Sim</i>
Banco de Dados	Descreve um banco de dados	<i>Sim</i>
Importe	Declara uma importação de outro arquivo	Não

construções – eventos de teste

Termo	Finalidade	Global
Antes de cada Cenário	Configurar o ambiente antes de cada cenário	Não
Depois de cada Cenário	Configurar o ambiente depois de cada cenário	Não
Antes da Funcionalidade	Configurar o ambiente antes de cada funcionalidade	Não
Depois da Funcionalidade	Configurar o ambiente depois de cada funcionalidade	Não
Antes de Todas	Configurar o ambiente antes de todas as funcionalidades	<i>Sim</i>
Depois de Todas	Configurar o ambiente depois de todas as funcionalidades	<i>Sim</i>

exemplo – busca.feature

#language: pt

Funcionalidade: Busca no Google

Cenário: Busca retorna resultado esperado

Variante: Busca ao teclar Enter

Dado que estou em "https://google.com.br"

Quando eu informo "concordialang.org" em <q>
e eu pressiono "Enter"

Então eu vejo "npm"

DSLs compatíveis com Gherkin

Funcionalidade: <título>

Como um <papel desempenhado>

Desejo <meta>

Para <benefício obtido>

História de Usuário (opcional)

Cenário: <título>

Dado que <contexto>

Quando <evento ocorre>

Então <resultado deve ocorrer>

Descrição de Cenário

exemplo

Funcionalidade: Ganhar milhas ao voar

Como um viajante

Desejo poder acumular milhas toda vez que voar

Para poder viajar de graça ou ter descontos

Cenário: Um novo viajante começa com nível Bronze

Dado que não sou um cliente da companhia Kaikai

Quando eu me cadastro no programa de milhas

Então eu recebo o nível Bronze

começando com uma Funcionalidade

#language: pt

Funcionalidade: Logout

indica a língua usada na especificação

não necessita de História de Usuário, se não acrescenta valor !

declarando um Cenário

#language: pt

Funcionalidade: Logout

Cenário: Vai para a página principal ao sair

Dado que estou logado no site

Quando eu aciono a opção de sair

Então sou redirecionado para a página principal do site

Cenário em linguagem não tecnológica

declarando uma Variante

#language: pt

Funcionalidade: Logout

Cenário: Vai para a página principal ao sair

Dado que estou logado no site

Quando eu aciono a opção de sair

Então sou redirecionado para a página principal do site

Variante: Sai quando aciona opção Sair

Quando eu clico em <Sair>

Então eu vejo a url "/"

Variante expressa interação com sistema

analisando de perto a Variante

#language: pt

Funcionalidade: Logout

Cenário: Vai para a página principal ao sair

Dado que estou logado no site

Quando eu aciono a opção de sair

Então sou redirecionado para a página principal do site

Variante: Sai quando aciona opção Sair

Quando eu clico em <Sair>

Então eu vejo a url "/"

também
chamado de
Literal de IU

<Sair> é um identificador de elemento da IU

analisando de perto a Variante

#language: pt

Funcionalidade: Logout

Cenário: Vai para a página principal ao sair

Dado que estou logado no site

Quando eu aciono a opção de sair

Então sou redirecionado para a página principal do site

Variante: Sai quando aciona opção Sair

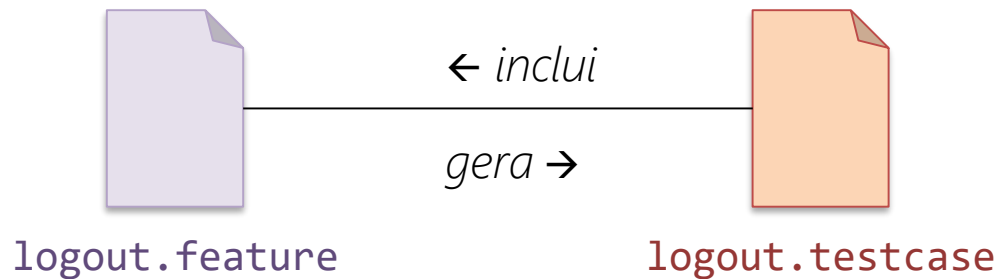
Quando eu clico em <Sair>

Então eu vejo a url "/"

"/" é um valor


gerando um caso de teste


```
$ concordia --just-testcase
```





caso de teste

#language: pt

importe "login.feature"  importa definições de Funcionalidades

@generated  tag que indica que o Caso de Teste foi gerado

@scenario(1)  tag que referencia Cenário pelo índice

@variant(1)  tag que referencia Variante pelo índice

Caso de Teste: Sai quando aciona opção Sair - 1

Quando eu clico em <Sair>

Então eu vejo a url "/"

Caso de Teste não
diferiu da Variante,
pois não houve
nada para gerar

voltando à Variante

#language: pt

Funcionalidade: Logout

Cenário: Vai para a página principal ao sair

Dado que estou logado no site

Quando eu aciono a opção de sair

Então sou redirecionado para a página principal do site

Variante: Sai quando aciona opção Sair

Quando eu clico em <Sair>

Então eu vejo a url "/"

Verificação não
está ocorrendo na
Variante.

Vamos incluir!

voltando à Variante

#language: pt

importe "login.feature"

importa definições de Login

Funcionalidade: Logout

Cenário: Vai para a página principal ao sair

Dado que estou logado no site

Quando eu aciono a opção de sair

Então sou redirecionado para a página principal do site

Variante: Sai quando aciona opção Sair

Dado que tenho ~usuário logado~

Quando eu clico em <Sair>

Então eu vejo a url "/"

~usuário logado~ é um estado do sistema

nesse caso,
produzido
por "Login"

em Login

#language: pt

Funcionalidade: Login

Cenário: Loga com sucesso

Dado que estou na página de login

Quando eu informo minhas credenciais

Então consigo acessar o sistema

Variante: Entra com credenciais de Administrador

Dado que estou em "/login"

Quando eu preencho <#usuario> com "admin"

e preencho <#senha> com "123456"

e cliço em <#entrar>

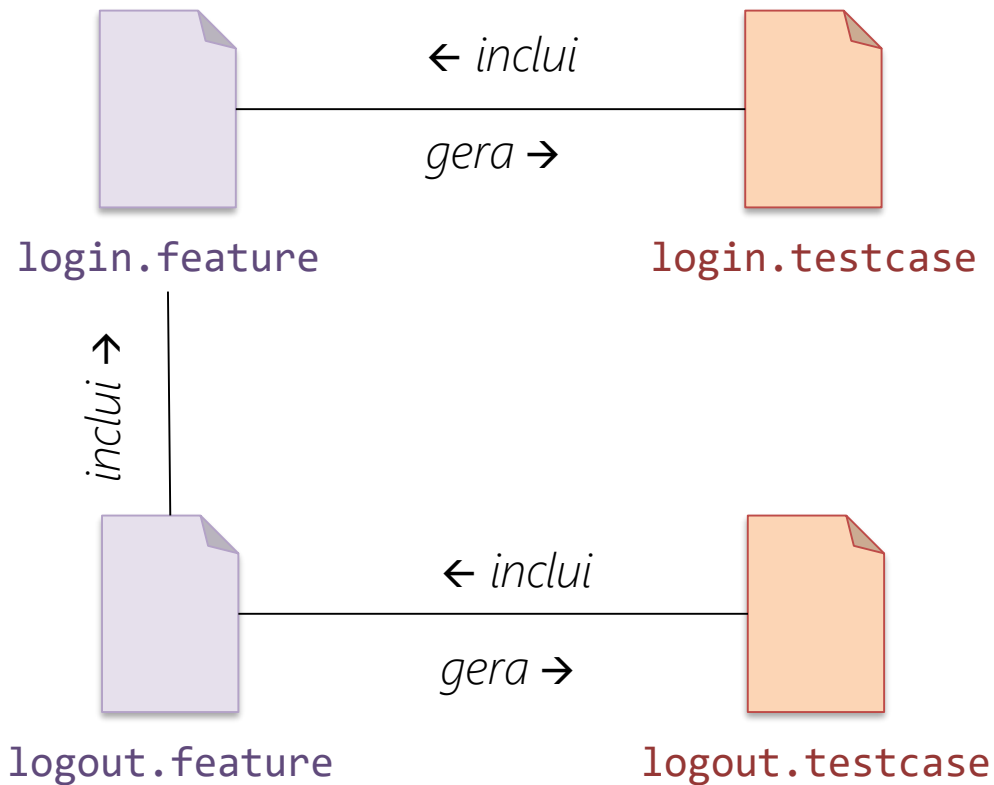
Então eu vejo "Olá"

e tenho ~usuário logado~

~usuário logado~ é produzido

regerando casos de teste

```
$ concordia --just-testcase
```



voltando ao caso de teste de Logout

```
#language: pt  
importe "login.feature"
```

```
@generated  
@scenario( 1 )  
@variant( 1 )
```

Caso de Teste: Sai quando aciona opção Sair - 1

Dado que estou em "/login"

Quando eu preencho <#usuario> com "admin"

e preencho <#senha> com "123456"

e cliço em <#entrar>

Então eu vejo "Olá"

Quando eu cliço em <Sair>

Então eu vejo a url "/"

Dado que tenho ~usuário logado~

estados do sistema

aparecendo em um **Dado que** denota uma **pré-condição**
necessidade que tenha executado antes

aparecendo em um **Quando** denota uma **chamada**
necessidade que seja executado agora

aparecendo em um **Então** denota uma **pós-condição**
estado é produzido

aparecendo em um **Dado que** denota uma pré-condição
necessidade que tenha executado antes

Dado que tenho ~usuário logado~

...

aparecendo em um **Quando** denota uma chamada
necessidade que seja executado agora

...

Quando tenho ~cidade cadastrada~

...

aparecendo em um **Então** denota uma pós-condição
estado é produzido

...

Então tenho ~usuário logado~

observações sobre estados

a ferramenta o **procura nos arquivos importados** quando um **estado** é referenciado, em **Dado que** ou **Quando**

se houver **mais de um produtor** do estado, mais de um Caso de Teste *pode* ser gerado

cada um Caso de Teste pode cobrir uma **combinação** diferente depende do **algoritmo** escolhido (opção da linha de comando)

definindo restrições para Login

...

Variante: Acesso com credenciais

Dado que estou em "/login"

Quando eu preencho {Usuario}, {Senha}

e clico em <#entrar>

Então eu vejo "Olá"

e tenho ~usuário logado~

Referência para Elementos de IU

Tabela

Tabela: Usuarios

usuario	senha
admin	123456
gerente	ger123

Elemento de IU: Usuario

Elemento de IU

- id é "#usuario"

- valor vem da consulta "SELECT usuario FROM [Usuarios]"

Caso contrário eu vejo "Usuário ou senha inválidos"

Restrição

Expectativa quando restrição ignorada

Elemento de IU: Senha

Elemento de IU

- id é "#senha"

- valor vem da consulta "SELECT senha FROM [Usuarios] WHERE usuario = {Usuario}"

Caso contrário eu vejo "Usuário ou senha inválidos"

Restrição

Expectativa quando restrição ignorada

representa um *widget* presente na tela

sintaxe comum:

Elemento de IU: <nome>

- <propriedade> <operador> <conteúdo>

 - [Caso contrário <sentença-1>]

 - [e <sentença-2>]

 - ...

 - [e <sentença-N>]

- . . .

por *default*

tem propriedade **tipo** com valor **textbox**

tem propriedade **tipo de dado** com valor **string**

tem propriedade **id** igual ao *nome*, convertido para *camel case*

por exemplo, **Elemento de IU: Nome Completo** é o mesmo que

Elemento de IU: Nome Completo

- id é "nomeCompleto"
- tipo é textbox
- tipo de dado é string

Elemento de IU - propriedades

- **id** é a identificação do elemento de IU na tela
- **tipo** é o tipo do elemento de IU
- **tipo de dado** é o tipo de dado aceito
- **editável** indica se pode ser editado pelo usuário
- **obrigatório** indica se é obrigatório
- **formato** define uma *expressão regular* como formato aceito
- **valor** define um valor para o elemento
- **valor mínimo** define um valor mínimo
- **valor máximo** define um valor máximo
- **comprimento mínimo** define um comprimento mínimo
- **comprimento máximo** define um comprimento máximo



sentença Caso contrário

estabelece o **comportamento esperado** quando um valor de entrada não **satisfaz** a **restrição** declarada

aceita para as propriedades

- obrigatório

- formato

- valor

- valor mínimo

- valor máximo

- comprimento mínimo

- comprimento máximo

equivale a uma sentença **Então**

tabela

define uma tabela de dados, consultável via SQL

tipo dos dados são inferidos

observação: datas devem estar no formato ano/mês/dia

sintaxe

Tabela: <nome>

	<coluna-1>		<coluna-2>		...		<coluna-K>	
	<valor-1>		<valor-2>		...		<valor-K>	
	...							
	<valor-1N>		<valor-2N>		...		<valor-KN>	

banco de dados

permite usar um arquivo ou banco de dados externo

atualmente Firebase, MySQL, PostreSQL, SQLite, MS Access, SQL Server, JSON, CSV, INI e Excel

sintaxe:

Banco de Dados: <nome>

- <propriedade> é "<valor>"



banco de dados - exemplos

Banco de Dados: Usuarios

- tipo é "json"
- caminho é "/caminho/ate/usuarios.json"

Banco de Dados: BD de Teste

- tipo é "mysql"
- nome é "testdb"
- usuário é "root"
- senha é ""



banco de dados - propriedades

uso **depende** do **banco de dados** utilizado

muitas assumem valores *default* do banco de dados
por exemplo, a **porta** padrão do MySQL é **3050**

propriedades

- tipo
- nome
- caminho
- hospedeiro (ou host)
- porta
- usuário
- senha
- opções



constantes

bloco de declarações, somente um por arquivo

sintaxe

Constantes:

- "<nome>" é <valor>
- "<nome-N>" é <valor-N>

exemplo

Constantes:

- "Login" é "/login"
- "PI" é 3.14159
- "Tamanho Mínimo" é 2



referências

para Elementos de IU

{Nome do Elemento} ou **{Funcionalidade:Nome do Elemento}**

para Constantes, Tabelas e Bancos de Dados

[Nome]

para Estados

~Estado~



Elementos de IU podem ser referenciados em

- Variantes

- Elementos de IU (propriedades)

- Consultas

Constantes podem ser referenciadas em

- Variantes

- Elementos de IU (propriedades)

- Consultas

- Bancos de Dados (propriedades)

Tabelas e Bancos de Dados podem ser referenciados em consultas

Estados podem ser referenciados em passos de **Variantes**

referências – exemplos

Banco de Dados: Usuarios

- tipo é "json"
- caminho é "usuarios.json"

Constantes:

- "Flag de Admin" é "A"
- "Cores" é ["Verde", "Azul", "Preto"]

Elemento de IU: Usuario

- valor vem de "SELECT usuario FROM [Usuarios] WHERE flag = [Flag de Admin]"

Elemento de IU: Cor do Tema

- valor está em [Cores]

casos de teste baseado em regras

required

REQUIRED_FILLED
REQUIRED_NOT_FILLED

format

FORMAT_VALID
FORMAT_INVALID

value is in <set or query>

SET_FIRST_ELEMENT
SET_RANDOM_ELEMENT
SET_LAST_ELEMENT
SET_NOT_IN_SET

minimum/maximum value

VALUE_LOWEST
VALUE_RANDOM_BELOW_MIN
VALUE_JUST_BELOW_MIN
VALUE_MIN
VALUE_JUST_ABOVE_MIN
VALUE_ZERO
VALUE_MEDIAN
VALUE_RANDOM_BETWEEN_MIN_MAX
VALUE_JUST_BELOW_MAX
VALUE_MAX
VALUE_JUST_ABOVE_MAX
VALUE_RANDOM_ABOVE_MAX
VALUE_GREATEST

minimum/maximum length

LENGTH_LOWEST
LENGTH_RANDOM_BELOW_MIN
LENGTH_JUST_BELOW_MIN
LENGTH_MIN
LENGTH_JUST_ABOVE_MIN
LENGTH_MEDIAN
LENGTH_RANDOM_BETWEEN_MIN_MAX
LENGTH_JUST_BELOW_MAX
LENGTH_MAX
LENGTH_JUST_ABOVE_MAX
LENGTH_RANDOM_ABOVE_MAX
LENGTH_GREATEST

CLI

instalação

```
npm install -g concordialang
```

instalando um plug-in

```
concordia --plugin-install codeceptjs
```

execução

iniciando o servidor de testes (só 1 vez)

```
concordia --plugin-serve codeceptjs
```

configurando um projeto

```
concordia --init
```

executando

```
concordia --plugin codeceptjs
```


exemplo – busca.feature

#language: pt

Funcionalidade: Busca no Google

Cenário: Busca retorna resultado esperado

Variante: Busca ao teclar Enter

Dado que estou em "https://google.com.br"

Quando eu informo "concordialang.org" em <q>
e eu pressiono "Enter"

Então eu vejo "npm"



EXPLORER

OPEN EDITORS

search.feature

EXAMPLE-EN

search.feature



search.feature x

```
1 Feature: Google Search
2
3 Scenario: Search returns expected result
4
5     Variant: Search content on pressing Enter
6         Given that I am on "https://google.com"
7         When I type "concordialang.org" in <q>
8         And I press "Enter"
9         Then I see "npm"
```

PROBLEMS

TERMINAL

1: cmd

C:\code\tmp\example-en>



MAVEN PROJECTS

exemplo – execução

`concordia --plugin codeceptjs`

irá gerar

`busca.testcase`

`test/busca.js`

e irá executar o teste `test/busca.js`

FUNCIONALIDADE



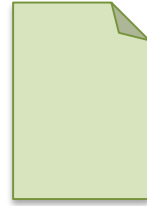
.feature

CASOS DE TESTE



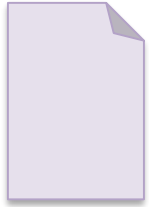
.testcase

SCRIPTS DE TESTE



.???

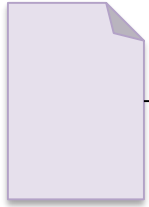
FUNCIONALIDADE



`login.feature`

exemplo

FUNCIONALIDADE



login.feature

CASOS DE TESTE



login.testcase

← inclui

gera →

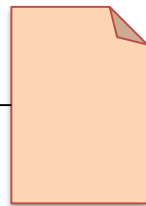
exemplo

FUNCIONALIDADE



login.feature

CASOS DE TESTE



login.testcase

SCRIPTS DE TESTE

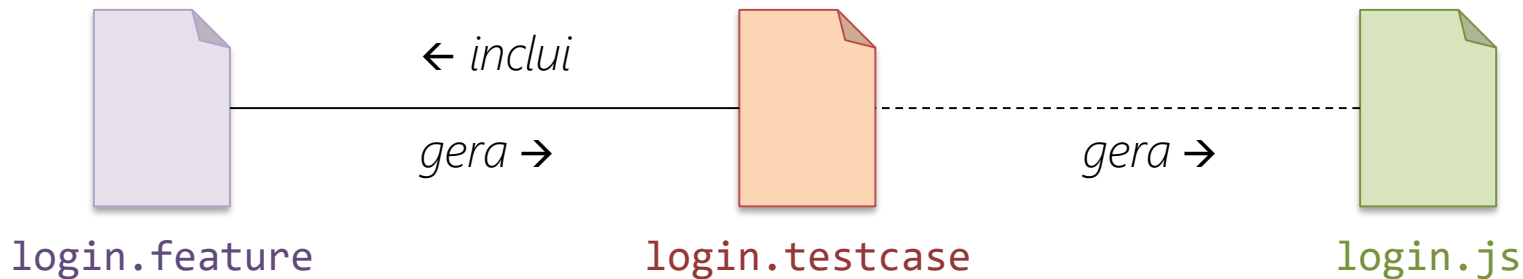


login.js

← inclui

gera →

gera →



exemplo

FUNCIONALIDADE

CASOS DE TESTE

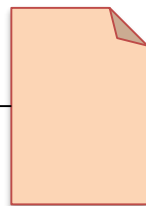
SCRIPTS DE TESTE



login.feature

← inclui

gera →



login.testcase

gera →



login.js



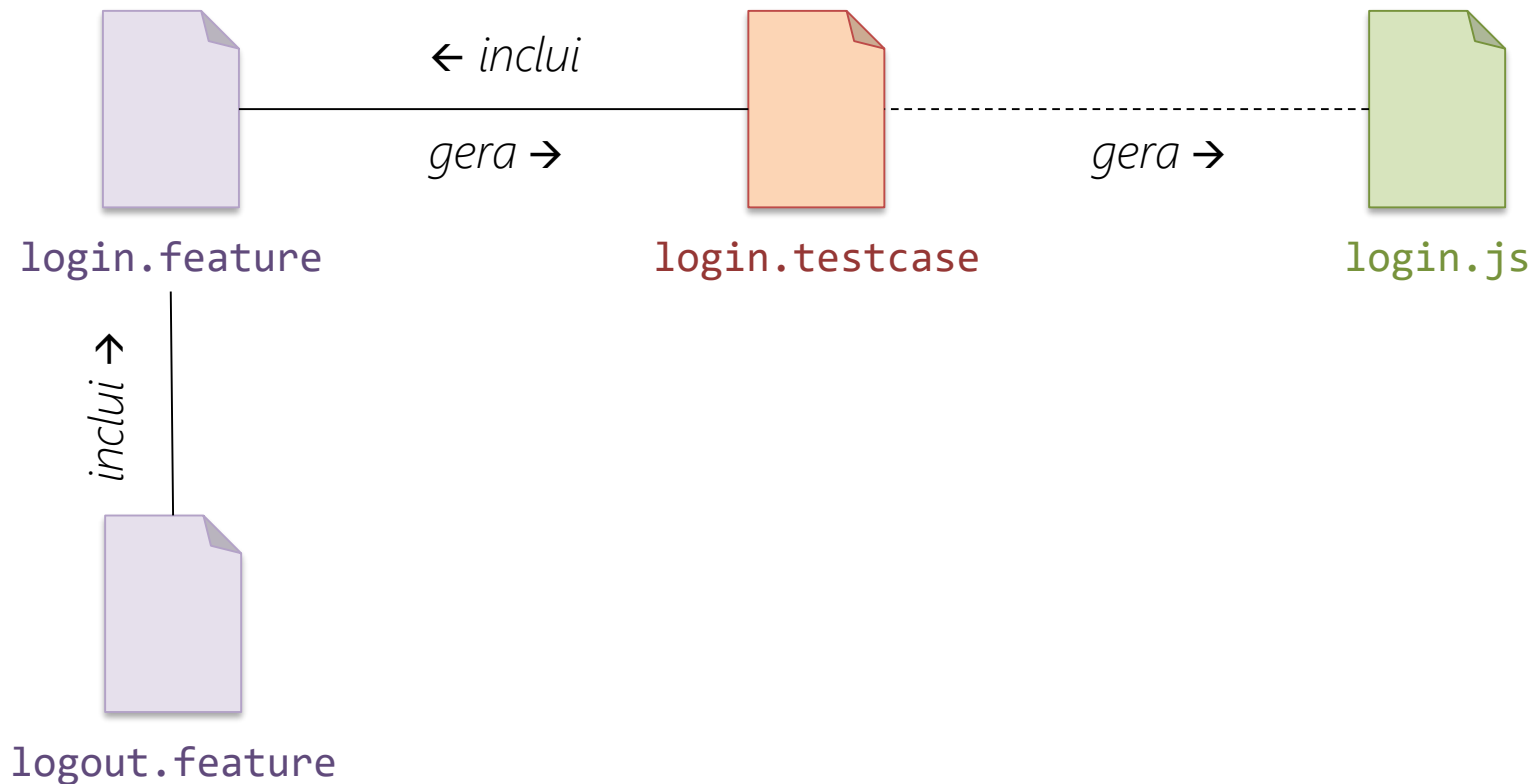
logout.feature

exemplo

FUNCIONALIDADE

CASOS DE TESTE

SCRIPTS DE TESTE

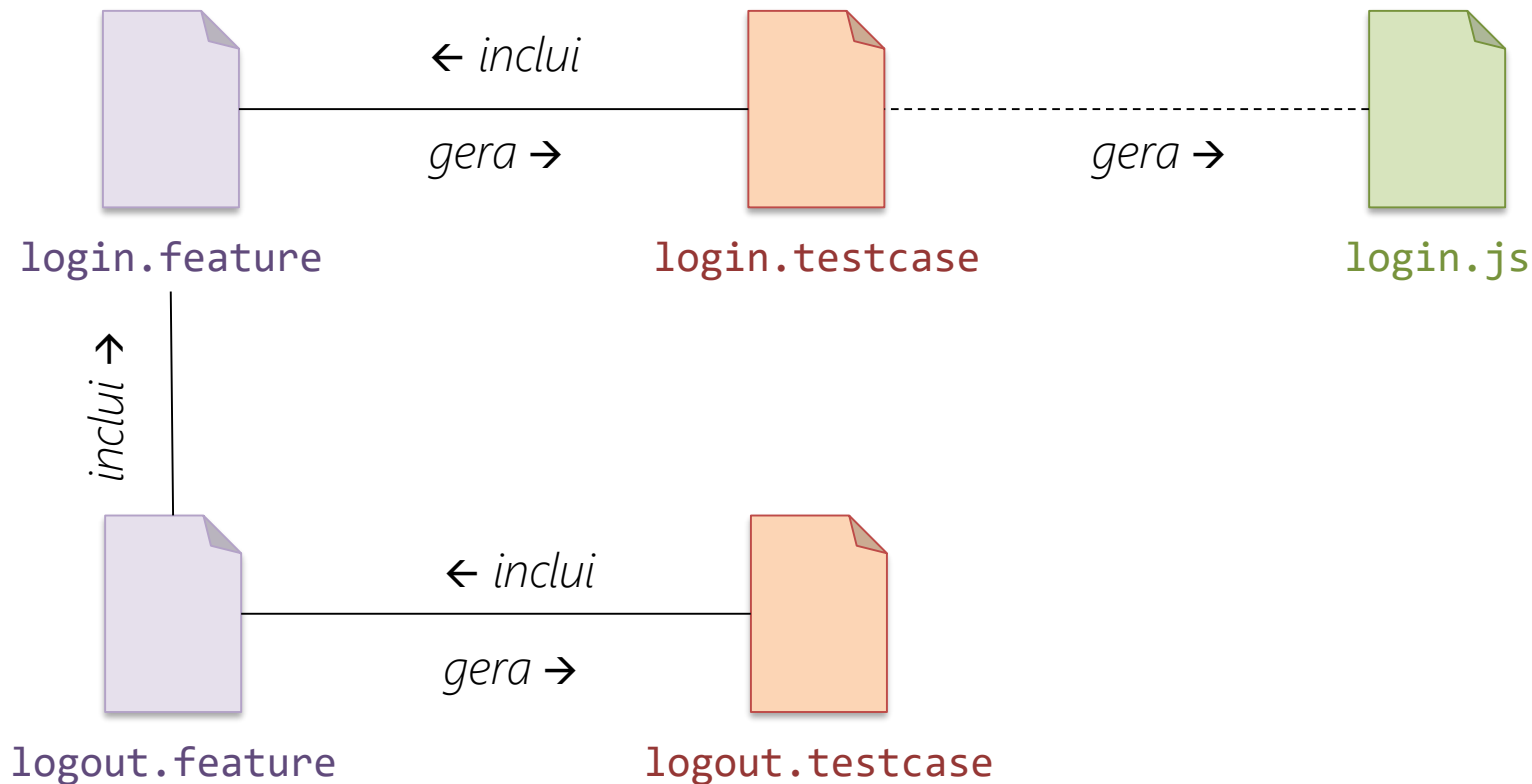


exemplo

FUNCIONALIDADE

CASOS DE TESTE

SCRIPTS DE TESTE

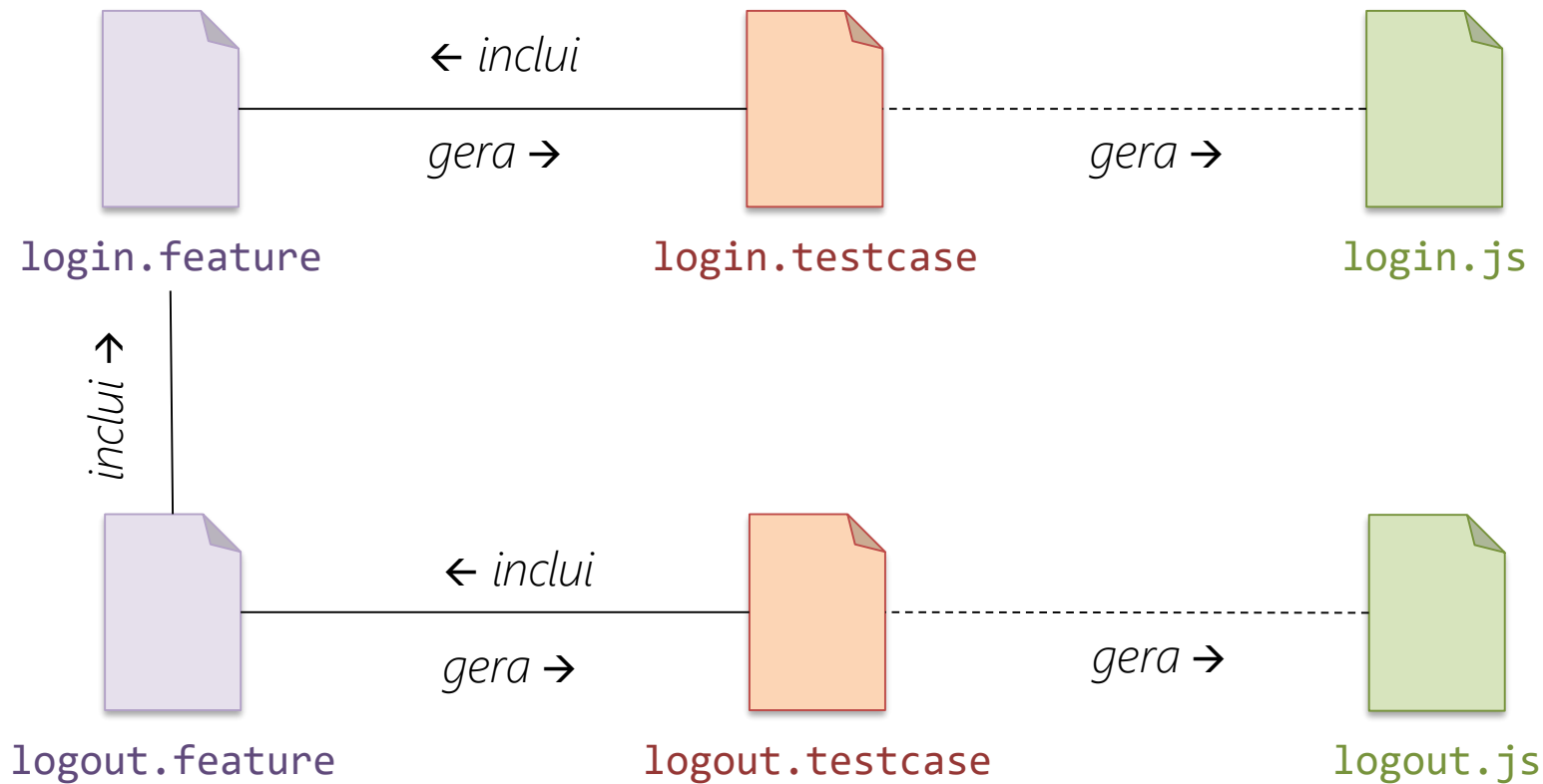


exemplo

FUNCIONALIDADE

CASOS DE TESTE

SCRIPTS DE TESTE



algumas opções

--plugin define o plugin a ser usado para gerar scripts de teste

concordia --plugin codeceptjs

--language definindo o idioma usado por padrão

concordia --language pt

--save-config salva um arquivo de configuração **.concordiarc** com os parâmetros da linha de comando informados

concordia --save-config --plugin codeceptjs

grava arquivo com plugin=codeceptjs e não será mais necessário informar esse parâmetro

`--dir-test-case` define o diretório de saída para Casos de Teste

concordia `--dir-test-case ./features`

`--dir-script` define o diretório de saída para scripts de teste

concordia `--dir-scripts ./test`

`--dir-result` define o diretório de saída para resultados de teste

concordia `--dir-result ./test`

algumas opções

- no-spec** deixa de processar a especificação
- no-test-case** deixa de gerar Casos de Teste
- no-script** deixa de gerar scripts de teste
- no-run** deixa de executar scripts de teste
- no-result** deixa de analisar resultados da execução dos testes

exemplo

```
concordia --plugin codeceptjs --no-run --no-result
```

algumas opções

- `--just-spec` só processa a especificação
- `--just-test-case` só gera Casos de Teste
- `--just-script` só gera scripts de teste
- `--just-run` só executa scripts de teste
- `--just-result` só analisa resultados da execução dos testes

exemplo

```
concordia --plugin codeceptjs --just-run
```

algumas opções

--case-ui indica o modo de string usado para gerar os ids de Elementos de IU quando não declarados. As opções são

camel – camelCase (default)

pascal – PascalCase

snake - snake_case

kabab - kebab-case

none - Does not change

exemplo

concordia --case-ui snake

algumas opções

-- **seed** indica uma string como **semente aleatória**

se **semente aleatória** for a **mesma**, serão produzidos os mesmos dados de teste aleatórios e os algoritmos com escolha aleatória produzirão os mesmos resultados

isso permite replicar uma geração, dando previsibilidade dos resultados

se não for informada, uma semente é gerada

sementes diferentes permitem variar dados de teste e caminhos de execução

exemplo

concordia --seed="minha semente"

--comb-variant indica o algoritmo de combinação de Variantes

se uma Variante precisa de um **estado** gerado por outra Variante, elas serão **combinadas**. Quando várias Variantes produzem o mesmo estado, a seleção delas pode ser feita por:

random – seleciona aleatoriamente (default)

first – seleciona a primeira Variante que produz o estado

fmi – seleciona a Variante mais importante, segundo a tag **@importance**

all – combina com todas

exemplo

```
concordia --comb-variant all
```

algumas opções

--comb-invalid indica quantos dados de entrada serão inválidos em cada teste com dados inválidos

se a aplicação valida um campo por vez, recomenda-se usar "1"

none – (ou **0**) nenhum dado inválido

1 – somente 1 dado inválido por vez

smart – deixa o algoritmo decidir (*default*)

random – número aleatório de dados inválidos por vez

all – todos os dados inválidos

exemplo

concordia --comb-invalid=1

algumas opções

--comb-data indica a combinação de casos de teste de dados

Cada Elemento de IU pode ter vários casos de teste aplicáveis. Combinar todos esses casos pode gerar muitos testes – bom antes de entrega de versão, mas demorado para testes frequentes.

sre – cada caso de teste aparece pelo menos uma vez (default)

sow – *one-wise* embaralhado

ow – *one-wise* fixo

all –todas as combinações

exemplo

concordia --comb-data=all

conclusões

possíveis usos para Concordia (1 de 3)

1. Especificar requisitos em mais de uma língua falada usando texto simples;
2. Validar requisitos com clientes;
3. Discutir requisitos e casos de teste entre a equipe de software (uso como mídia de comunicação);
4. Especificar casos de teste funcionais em linguagem natural (restrita);
5. Checar erros sintáticos, semânticos e lógicos em especificações;

6. Gerar, executar e analisar casos de teste e scripts de teste funcional completos;
7. Usar fontes de dado externas, como bancos de dados, para criar restrições sobre elementos de IU e produzir casos de teste;
8. Descobrir defeitos, especialmente em aplicações recentes;
9. Verificar a correspondência de uma aplicação com sua especificação Concordia;
10. Apoiar a adoção de BDD, ATDD e SbE;

11. Apoiar a adoção de testes funcionais em aplicações novas ou legadas;
12. Suportar *test-driven maintenance*;
13. Usar uma abordagem de manutenção orientada a modificação dos requisitos – isso é, muda os requisitos, usa a ferramenta para produzir os respectivos testes e então modifica a aplicação para passar nos testes;
14. Separar declarações de negócio das declarações em nível de teste;
15. Definir eventos de teste em linguagem natural (restrita) para configurar o estado de aplicações antes ou depois da execução de scripts de teste.

Katalon Concordia

<https://github.com/thiagodp/katalon-concordia>

Geração de Protótipos de IU

Trabalho de Conclusão de Curso – Pablo Veiga e Willian Gonçalves

Detecção do Impacto de Modificações para Regeneração de Testes

Trabalho de Conclusão de Curso – Gabriel Knupp

Novo Plug-in visando Teste de Aplicações de Linha de Comando

Trabalho de Conclusão de Curso – Danilo Striotto

futuro

muitas novidades vindo por aí...



perguntas