

Tutorial

BDD plus JUnit5

# Residente: Rosane Caldeira Data: 10/10/2022

## Passo 1: Motivação

<https://blog.onedaytesting.com.br/gherkin/>

Não é novidade que a modalidade “teste automatizado” tem crescido rapidamente no mercado de qualidade de software, mas já se perguntou o porquê?

O teste automatizado automatiza rotinas repetitivas de teste para que o testador não precise realizar todo o fluxo de teste manualmentee economize tempo de modo que possa ser utilizado para focar em outras tarefas mais específicas.

Diante deste cenário surge a seguinte pergunta: onde o **“Gherkin”** entra nisso?

O Gherkin é um dos elementos principais quando se trata de **Behaviour-Driven Development (BDD) em automação**. Sua função é padronizar a forma de descrever especificações de cenários, baseado na regra de negócio. Muitas vezes, o Gherkin é confundido com o BDD. Por isso é bom lembrar que a maneira como escrevemos os cenários não é BDD.

O Gherkin é uma linguagem natural que pode ser compreendida desde o time de negócio até o time técnico e segue a conhecida estrutura Given, When, Then. Essa é a grande vantagem: com ele criamos uma documentação que será usada e compreendida por todos, tanto pessoas quanto máquinas, e quando integrado com uma ferramenta como o [**Cucumber**](https://cucumber.io/), por exemplo, pode-se aplicar a automatização e integração com o [JUnit 5](https://junit.org/junit5/) via dependências providas por ferramentas de gerenciamento de projeto, tais como: [Apache Maven](https://maven.apache.org/) debaixo de dois ambientes de desenvolvimento que sejam: [Visual Studio Code](https://code.visualstudio.com/) e/ou [Eclipse](https://www.eclipse.org/)

<https://blog.onedaytesting.com.br/bdd-introducao/>

A ideia do BDD surgiu a partir dos desafios que Dan North enfrentava com a prática do TDD. Ele começou a perceber que, na maioria das vezes, os desenvolvedores escreviam nomes de testes que não eram nem um pouco claros e não demonstravam o que realmente o teste deveria fazer, e, por conta disso, outras pessoas da equipe e às vezes até mesmo quem escreveu os testes, não conseguia entender sobre o que aquilo se tratava.

Foi quando ele percebeu que o simples fato de mudar o nome do método para algo mais completo e focado no comportamento e acrescentando somente a palavra ‘should’, fez toda a diferença, tornando o teste mais fácil de manter por ter um objetivo mais claro. Com isso, Dan North passou a usar a palavra comportamento ao invés de teste.

Portanto, o BDD, é uma metodologia que envolve uma série de práticas que apoiam times para entregas de qualidade e de forma mais ágil, sempre com foco no comportamento que o software deve possuir. O objetivo principal do BDD é auxiliar e melhorar a comunicação do time, e desta forma permite construir *features* com valor real para o negócio.

Algumas vantagens ao utilizar o BDD:

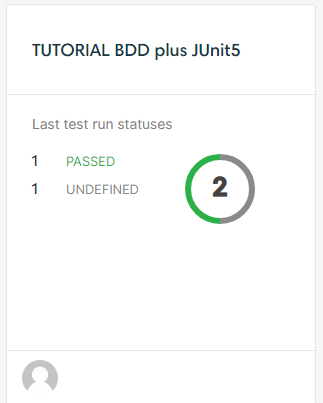
* redução de desperdício
* redução de custos
* mudanças mais fáceis e seguras
* atualizações mais rápidas

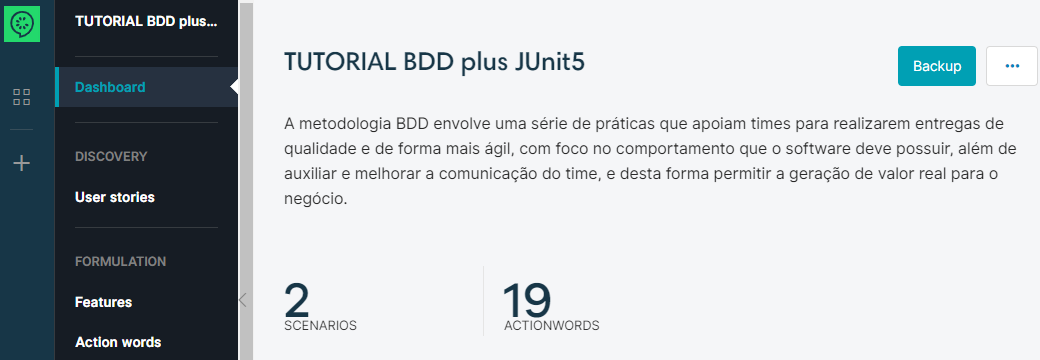
Como John Ferguson coloca em seu livro ‘**BDD in Action**’: o BDD é baseado em alguns simples princípios:

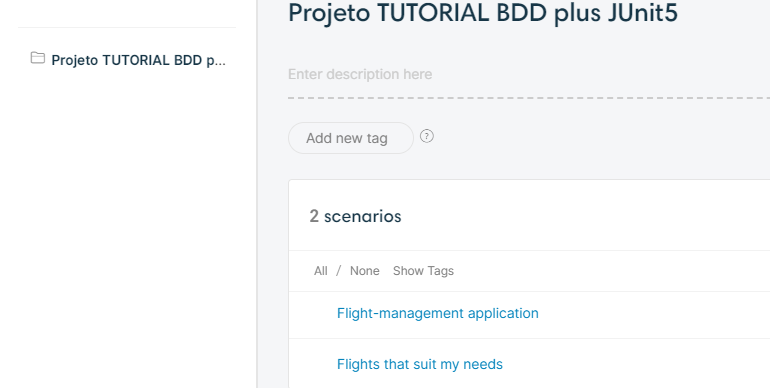
* Descrever comportamentos, não especificar soluções;
* Descobrir o comportamento que vai entregar um real valor de negócio;
* Usar conversas e exemplos para explorar o que um sistema deveria fazer.

## Passo 2: preparação do ambiente - baixar, instalar, configurar para executar seu projeto

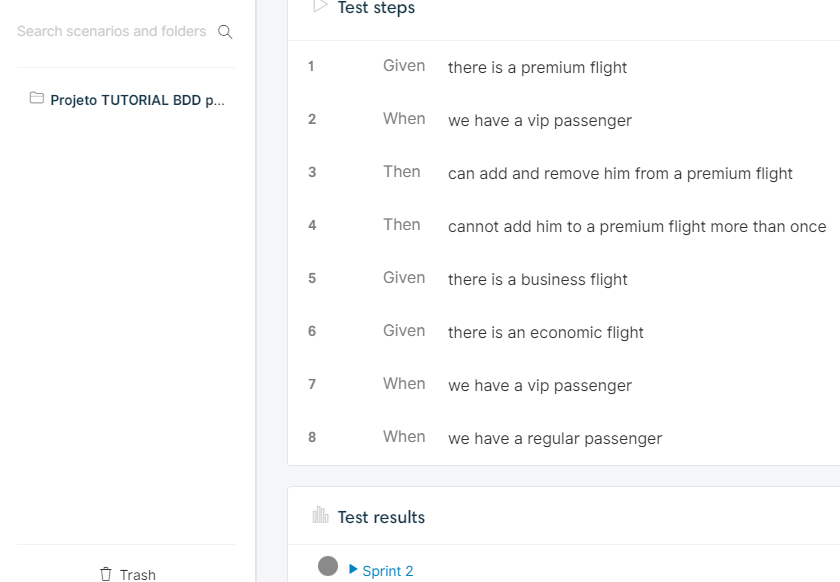
* [**Cucumber**](https://cucumber.io/)

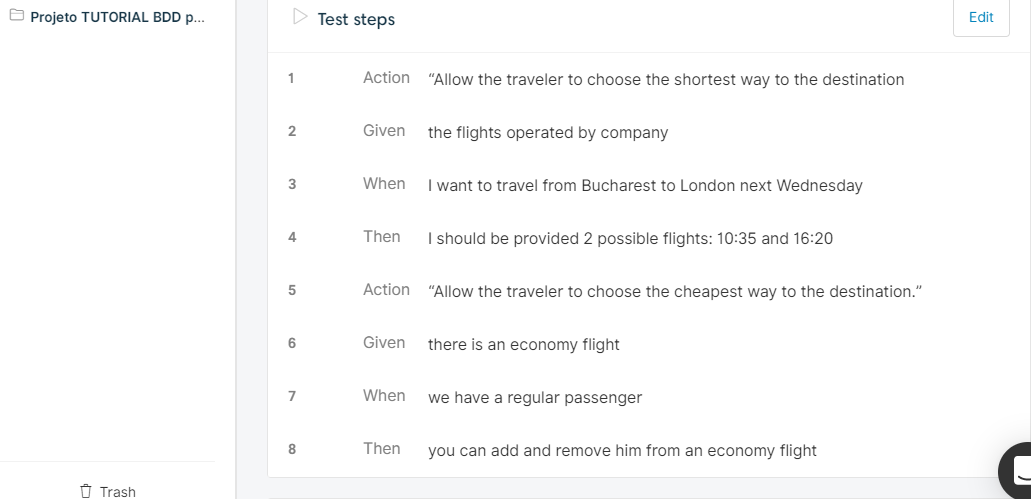
****

**Feito com base na literatura de Catalin Tudose In Action BDD - Working BDD style with Cucumber and JUnit 5 **

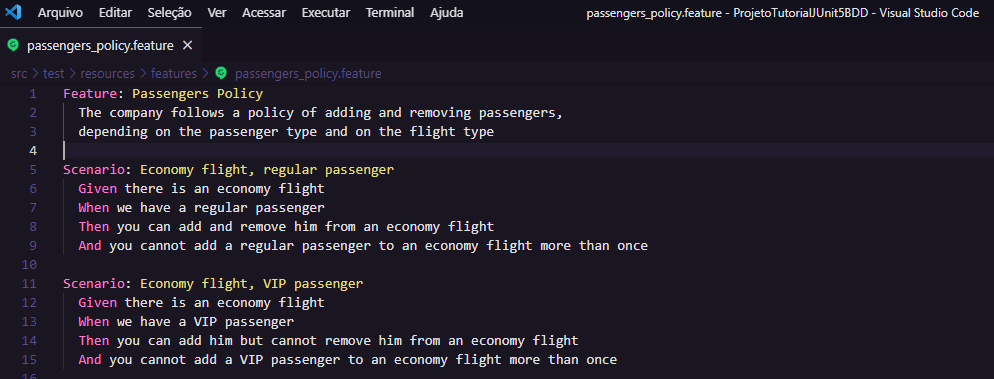
****

**Projeto -> cenários -> features -> escritos no cucumber**

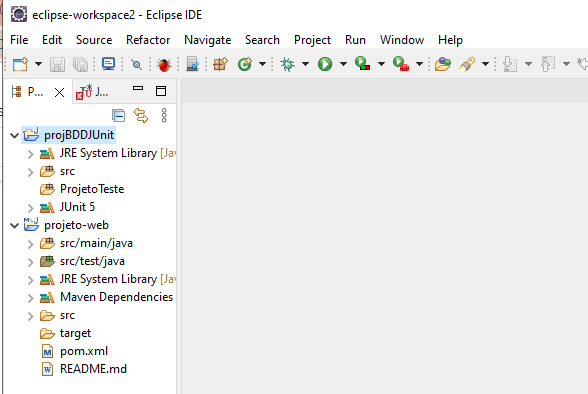
****

****

* [**Visual Studio Code**](https://code.visualstudio.com/)

****

* [**Eclipse**](https://www.eclipse.org/)

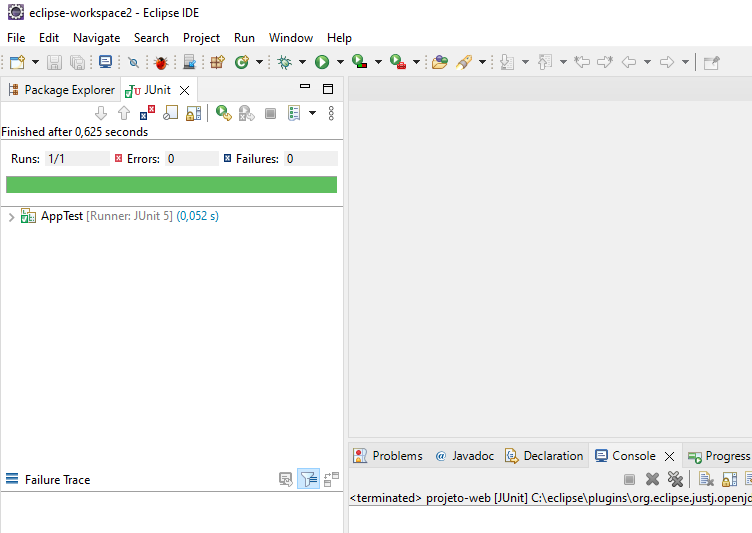
****

* [**Apache Maven**](https://maven.apache.org/)
* **Dependências e plugins:**

Atenção especial ao arquivo pom.xml que carrega as dependências necessárias para pronta execução do seu projeto Maven no Eclipse respeitando as regras sintáticas de elicitação requisitos para a condução do projeto.

* + [**https://giuliana-bezerra.medium.com/criando-um-projeto-java-maven-no-eclipse-cf7326d4db37**](https://giuliana-bezerra.medium.com/criando-um-projeto-java-maven-no-eclipse-cf7326d4db37)

## Passo 3: execução do seu projeto

****

## Passo 4: conclusão

Como pudemos ver, o BDD está ligado a todo o processo de desenvolvimento, com o objetivo de facilitar a comunicação através de conversas estratégicas, além de oferecer práticas que proporcionam um melhor entendimento das funcionalidades em questão. Com foco no comportamento, o BDD está diretamente ligado a geração de valor, tudo que é pensado e discutido deve atender a um bem maior que é criar uma funcionalidade que gerará um resultado imprescindível ao produto e seus usuários.

## Referências, pesquisas, frases e trabalhos futuros

Smart, John Ferguson. BDD IN ACTION, 2014.

Tudose, Catalin. JUnit IN ACTION, 2020.

[Hiptest · GitHub](https://github.com/hiptest)

[Getting started with Behavior Driven Development](https://cucumber.io/tools/cucumberstudio/getting-started-with-behavior-driven-development/)

[CucumberStudio | #1 BDD Collaboration Tool](https://cucumber.io/tools/cucumberstudio/ci-in-5-minutes-flat/)

<https://github.com/hiptest/hps-cucumber-java>

[Começando com Apache Maven em projetos Java](https://www.youtube.com/watch?v=2K_DhuDK5BA)

<https://spaceprogrammer.com/java/estrutura-basica-de-um-projeto-web-no-eclipse/#:~:text=Abra%20o%20arquivo%20e%20no,Junit%20j%C3%A1%20vem%20como%20padr%C3%A3o>.

<https://giuliana-bezerra.medium.com/criando-um-projeto-java-maven-no-eclipse-cf7326d4db37>

<https://www.youtube.com/channel/UCxaSvucGLNutBeK1vMiF6Yw>

*O know-how da Sofist consegue nos levar a outro nível de consciência quando falamos em teste de carga. Desde a flexibilidade na montagem dos scripts de testes até o entendimento profundo que ela tem do nosso negócio, ela vem se provando ano a ano um parceiro de valor para nossos resultados estratégicos”.*

<https://www.youtube.com/watch?v=fRuAz6B9YoE>

Thiago (Jesus) Catoto – Diretor de Tecnologia no Magazine Luiza

*“Ignorance is the single greatest impediment to throughput” –*

Dan North

A intenção é dar continuidade as pesquisas e, sobretudo, práticas voltadas para a integração de técnicas e tecnologias de programação, essenciais para uma boa programação de computadores.

O maior desafio deste projeto foi encontrar as dependências apropriadas assim como plugins das ferramentas. Surgiram restrições no momento da execução