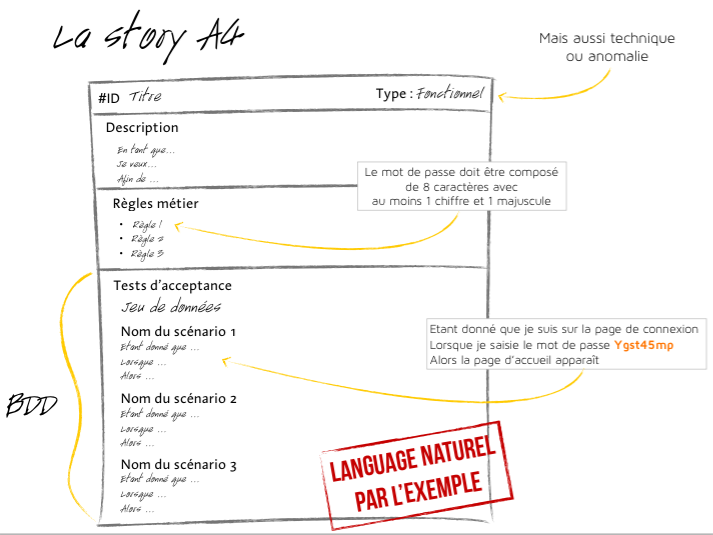
Comment traduire les scenarios de tests en Gherkin?



1. Grouper les cas de tests par une fonctionnalité global domaine et introduire sa description.

**Feature**: checkout

In order to ...

As a ...

I need to ...

**Scenario**: try checkout

1. Adapter ces cas de tests en scenarios et aux exemples qui décrivent un comportement dans le perimetre de cette fonctionnalité. Un scenario est un cas d’utilisation spécifique de cette fonctionnalité. Les scenarios devrait être indépendant l’un de l’autre et couvrir qu’un seul comportement.
2. Un scenario contient des instructions de haut niveau:

Given that user is logged in with <name> and <password>

au lieu de

Given that user types username

And types the password

And clicks on submit

1. Givens set up initial state, Whens perform an action, and Thens verify outcomes Given
2. Il est mieux d’ecrire les actions en troisième personne.
3. Une bonne pratique c’est parametriser le mot-cles
4. Pour couvrir plusieurs variantes du meme comportement utiliser le Scenario Outline
5. Design your keywords already parametrized

<https://automationpanda.com/2017/01/30/bdd-101-writing-good-gherkin/>

lengthy end-to-end tests

**Problèmes:**

1. Répétitions des pas de tests lourdes peut provoquer le rallongement du temps d’exécution. Tester en parallèle peut entraîner la corruption de données, mais l’exécution peut être lancée dans les tests suites différents.

If you have a long and performance heavy steps in your test, like the login with authentication, then separating tests limitlessly will cause your automation time to be too long to worth it

The reason to separate the behaviors is for understanding them, communicating them, and covering them. Yes, there is a bit of a performance hit, but running tests in parallel with an optimized framework makes it not so painful, and the test results are MUCH easier Ron triage and explain.

<https://automationpanda.com/2017/10/14/bdd-101-unit-integration-and-end-to-end-tests/>

1. @Before hook est globale une fois écrit dans une classe est utilisé partout et son exécution affect les résultats de tests.

The best strategy is to use global hooks only when necessary for time-intensive setup that can be shared safely. Any shared test data should be immutable. Always question the need for global hooks. Most tests probably won’t need them.

<https://automationpanda.com/2017/03/03/cucumber-jvm-global-hook-workarounds/>

1. Les entrants sont inscrit dans le script
2. Pas de possibilité de réutiliser des scenarios ou des features as Given, il faut utiliser Background
3. Picocontainer peut être utilisé pour injecter un driver entre des mots-clés implémentés dans des classes différentes. PageObject

mistakes <https://www.spritecloud.com/the-3-most-common-mistakes-writing-gherkin-features/>

Acceptance tests can come in different levels of granularity. Most of the time they will be rather high-level and test your service through the user interface. However, it's good to understand that there's technically no need to write acceptance tests at the highest level of your test pyramid. If your application design and your scenario at hand permits that you write an acceptance test at a lower level, go for it. Having a low-level test is better than having a high-level test. The concept of acceptance tests - proving that your features work correctly for the user - is completely orthogonal to your test pyramid.

## **RestAssured**

RestAssured is an API automation tool for Java.

In Cucumber-JVM calling steps from step definitions is not supported; this is by design.

The best way to achieve composition and reuse, is to use the features of your programming language. If you want to combine several actions into one step, extract individual (helper) methods and call these methods from your step definition.