

# Tests d'acceptation automatisés avec Specflow



# Bio

Feature: Confoo Speaker

*Present a talk at Confoo*

@Confoo2022

Scenario: Speaker shows a slide about his background

Given A talk at Confoo

And Talk has begun

When Slide is Bio

Then Speaker talks about his background

And Speaker picture should be shown

And LinkedIn profile should be displayed as <https://www.linkedin.com/in/frgauthier/>



<https://www.linkedin.com/in/frgauthier/>

# done.

votre partenaire techno

une division PYXIS

# done.

votre partenaire techno

## NOS SERVICES

- Développement logiciel sur mesure
- Extension d'équipe TI
- Formation et coaching technique



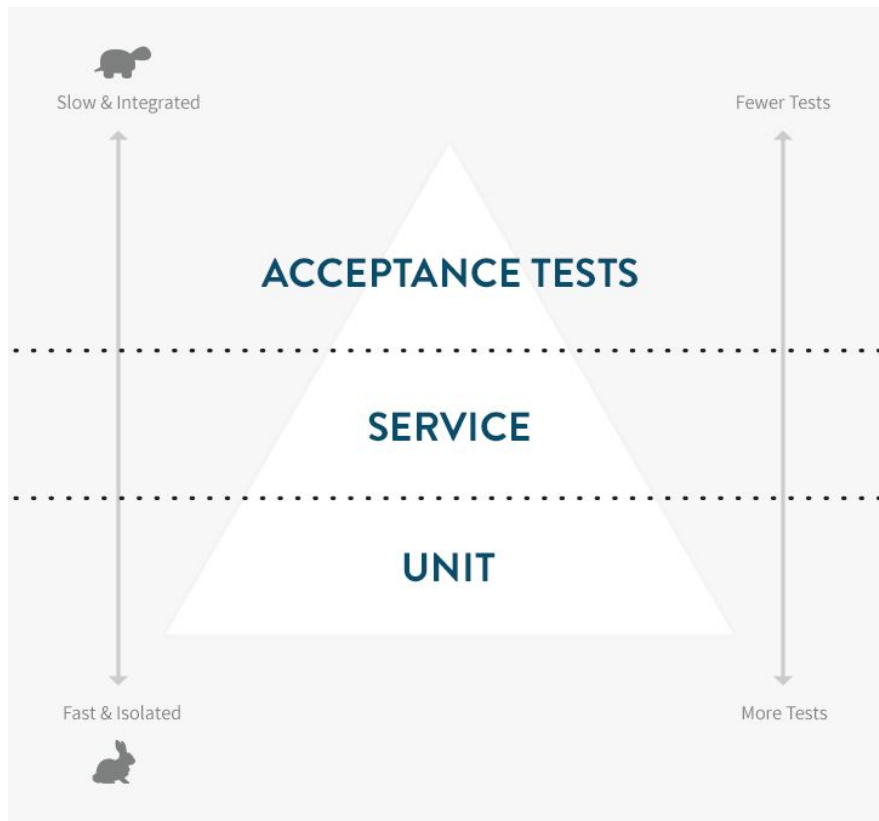
Notre équipe est toujours à la recherche de nouveaux projets intéressants.

Nos développeurs et conseillers experts contribuent à la création rapide de valeur et à l'atteinte des objectifs de croissance tout en maximisant le potentiel des gens.

[donetechno.com](https://donetechno.com)

# Type de tests

Automatisation?

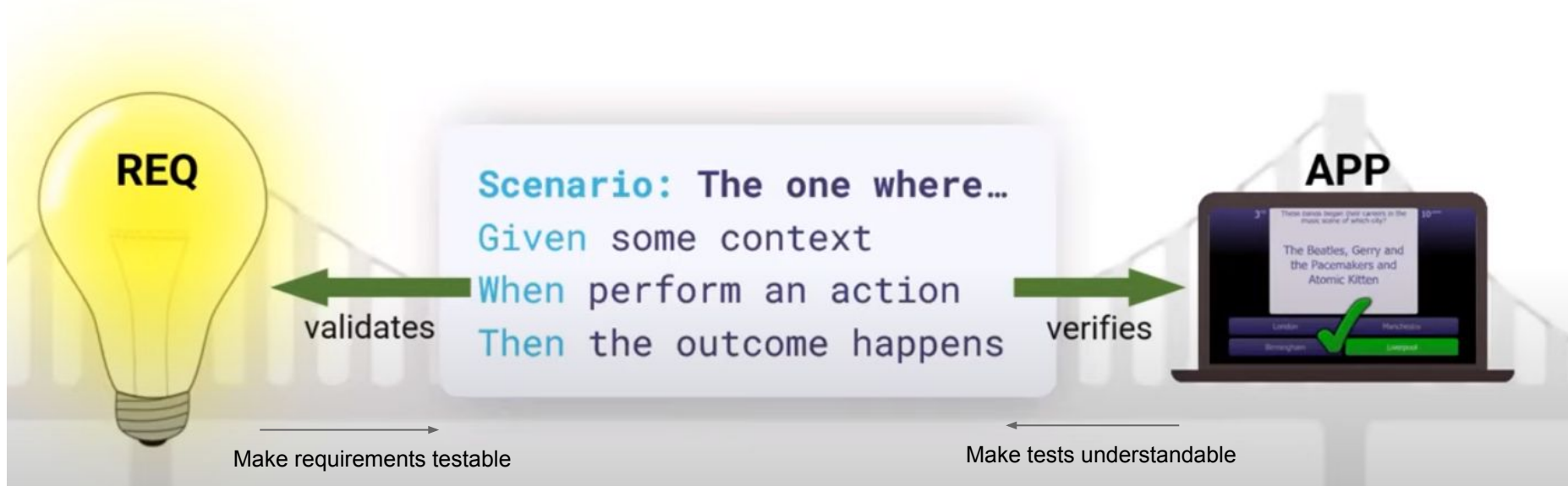


SpecFlow!

# Tests d'acceptance

- Vérifie si le système est conforme au comportement attendu
- Basé sur:
  - Les user stories
  - Critères d'acceptations
  - Des exemples (cas de tests)
- Écrit dans un langage naturel nommé Gherkin
- Partie automatisée du Behavior Driven Development

# Behavior Driven Development



Source: Lesson 1: Practicing BDD with SpecFlow  
<https://youtu.be/bgaWqgRTYWY>

# 3 Phases du BDD

## Discovery

Shared understanding is established through collaboration and structured conversations

## Formulation

The examples of system behaviour are documented as scenarios

## Automation

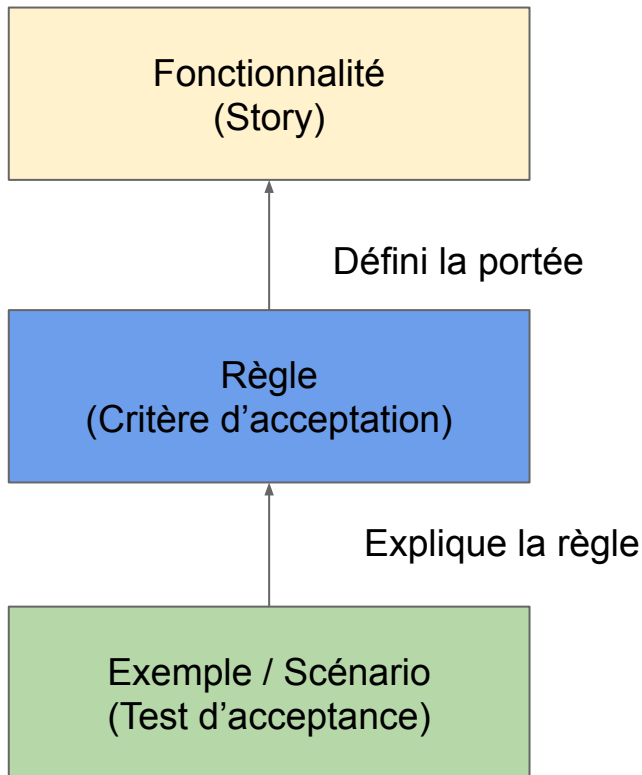
Scenarios are automated to be able to verify the behaviour of the system

Source: Lesson 1: Practicing BDD with SpecFlow

<https://youtu.be/bgaWqgRTYWY>



# Découverte d'un scénario



# Formulation : Langage Gherkin

## ▽ **Scenario:** Some determinable business situation

**Given** some precondition

**And** some other precondition

**When** some action by the actor

**Then** some testable outcome is achieved

**And** something else we can check happens too

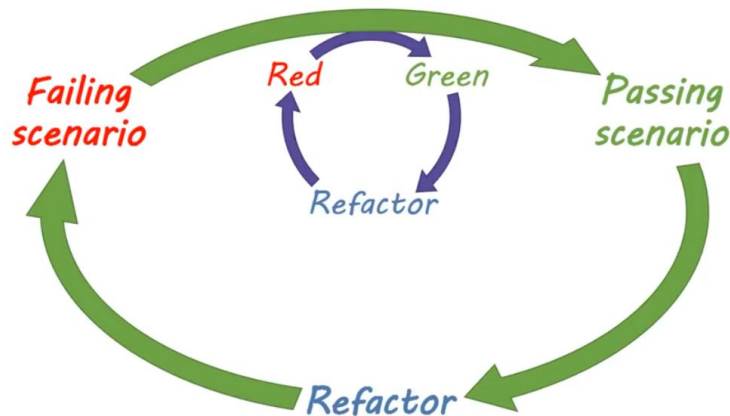
Given-When-Then can be also compared to the "Triple-A" testing pattern which is well known in unit testing:

Given-When-Then
Given
When
Then

Triple A testing pattern
Arrange
Act
Assert

# Automatisation: Test Driven

- Écrire les scénarios avant le code
- Même cycles que TDD:
  - RED
  - GREEN
  - REFACTOR
- Automatisation
  - SpecFlow
  - CI, Pipeline Devops



Source: Lesson 1: Practicing BDD with SpecFlow

<https://youtu.be/bgaWqgRTYWY>

# Specflow

## Feature: FizzBuzz

*Write a program that prints the numbers from 1 to 100.*

*But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz".*

*For numbers which are multiples of both three and five print "FizzBuzz".*

@tag1

### Scenario: Print FizzBuzz

Given the number is <number>

When I print the result

Then <FizzBuz> should be printed

### Examples:

number	FizzBuz
1	1
2	2
3	Fizz

The screenshot shows a test runner interface with a toolbar at the top containing icons for running, debugging, and other test-related actions. Below the toolbar, a tree view displays the test hierarchy. The root node is 'Test', which expands to show 'Confoo2022.SpecFlow.Calculator.Specs (1)'. This node expands to 'Confoo2022.SpecFlow.FizzBuzz.Specs (100)', which further expands to 'Confoo2022.SpecFlow.FizzBuzz.Specs.Features (100)'. This node expands to 'FizzBuzzFeature (100)', which finally expands to 'PrintFizzBuzz (100)'. Under 'PrintFizzBuzz (100)', there is a list of 15 test cases, each marked with a green checkmark, indicating they all passed. The test cases are: 'Print FizzBuzz(number: "1", fizzBuz: "1", exampleTags: [])', 'Print FizzBuzz(number: "10", fizzBuz: "Buzz", exampleTags: [])', 'Print FizzBuzz(number: "100", fizzBuz: "Buzz", exampleTags: [])', 'Print FizzBuzz(number: "11", fizzBuz: "11", exampleTags: [])', 'Print FizzBuzz(number: "12", fizzBuz: "Fizz", exampleTags: [])', 'Print FizzBuzz(number: "13", fizzBuz: "13", exampleTags: [])', 'Print FizzBuzz(number: "14", fizzBuz: "14", exampleTags: [])', and 'Print FizzBuzz(number: "15", fizzBuz: "FizzBuzz", exampleTags: [])'.

Test

- Confoo2022.SpecFlow.Calculator.Specs (1)
  - Confoo2022.SpecFlow.FizzBuzz.Specs (100)
    - Confoo2022.SpecFlow.FizzBuzz.Specs.Features (100)
      - FizzBuzzFeature (100)
        - PrintFizzBuzz (100)
          - Print FizzBuzz(number: "1", fizzBuz: "1", exampleTags: [])
          - Print FizzBuzz(number: "10", fizzBuz: "Buzz", exampleTags: [])
          - Print FizzBuzz(number: "100", fizzBuz: "Buzz", exampleTags: [])
          - Print FizzBuzz(number: "11", fizzBuz: "11", exampleTags: [])
          - Print FizzBuzz(number: "12", fizzBuz: "Fizz", exampleTags: [])
          - Print FizzBuzz(number: "13", fizzBuz: "13", exampleTags: [])
          - Print FizzBuzz(number: "14", fizzBuz: "14", exampleTags: [])
          - Print FizzBuzz(number: "15", fizzBuz: "FizzBuzz", exampleTags: [])

# Exemple & Comparaison

## Test unitaire (Triple A's)

```
[Fact]
0 references
public void ShouldCalculateSumOfTwoNumbers()
{
    //arrange
    var calculator = new Calculator
    {
        FirstNumber = 50,
        SecondNumber = 70
    };

    //act + assert
    calculator.Add().Should().Be(expected: 120);
}

[Fact]
0 references
public void ShouldCalculateSubstractionOfTwoNumbers()
{
    //arrange
    var calculator = new Calculator
    {
        FirstNumber = 120,
        SecondNumber = 70
    };

    //act + assert
    calculator.Subtract().Should().Be(expected: 50);
}
```

## Test d'acceptance (Specflow)

Feature: Calculator  
![Calculator](https://specflow.org/wp-content/uploads/2020/09/calculator.png)  
In order to avoid silly mistakes  
As a math idiot  
I \*want\* to be told the **sum** of **two** numbers

@mytag  
Scenario: Add two numbers  
Given the first number is 50  
And the second number is 70  
When the two numbers are added  
Then the result should be 120  
  
Scenario: Subtract two numbers  
Given the first number is 120  
And the second number is 70  
When the two numbers are subtracted  
Then the result should be 50

# SpecFlow - Feature File

tags

```
@delivery
Feature: Customers can collect their orders _____ feature name

Customers can choose to collect their order from the restaurant:
- whether they are authenticated or not _____ free-text description
- whether they pay on order or pay on collection

Rule: Customer should be informed when their order will be ready for customer-collection _____ rule grouping (optional)

# consider assuming an authenticated customer by default _____ comment
@important
Scenario: Estimated time of order completion is displayed _____ scenario name

steps
- Given a customer is authenticated
- And the customer placed an order _____ step conjunction (continuation)
  | time of order | preparation time | _____ data table
  | 18:00         | 0:30         |
- When they choose to collect their order
- Then the estimated time of order completion should be displayed as 18:30
```

Source: BDD Books - Formulation, <http://bddbooks.com/resources/formulation/figures/>

# SpecFlow - Feature File (avancé)

`#language: en` — feature file language specifier

`Feature: Blocklisting`

`Background:`

`Given a customer is authenticated`

Background – shared steps that are included in every scenario within the same file

`Rule: Customer's email OR phone must match those on blacklist`

`Scenario Outline: Decide if customer is restricted`

Scenario Outline – a template for generating scenarios

`Given the contact details are on the blacklist:`

`| email | phone number |`  
`| <email listed> | <phone listed> |`

parameter placeholders

`When the blacklist is checked`

`Then customer should be treated as <status>`

`Examples:`

column for documenting the example

parameter names

<code>description</code>	<code>email listed</code>	<code>phone listed</code>	<code>status</code>
<code>Both listed</code>	<code>yes</code>	<code>yes</code>	<code>restricted</code>
<code>Email listed</code>	<code>yes</code>	<code>no</code>	<code>restricted</code>
<code>Phone listed</code>	<code>no</code>	<code>yes</code>	<code>restricted</code>

columns containing data variations

`@fast` — tagged examples

`Examples: Unrestricted cases`

<code>description</code>	<code>email listed</code>	<code>phone listed</code>	<code>status</code>
<code>Neither listed</code>	<code>no</code>	<code>no</code>	<code>unrestricted</code>

examples  
(data rows)

MORE VIDEOS

# SpecFlow+LivingDoc

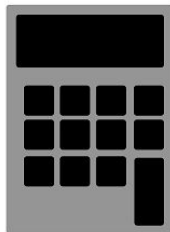
Living Documentation Test Result Summary

Filter by Keyword Filter by X

+ -

- SpecFlowCalculator.Specs 1 Passed 0 Failed 0 Others
  - Features 1 Passed 0 Failed 0 Others
    - Calculator
      - Add two numbers

## Feature: Calculator



In order to avoid silly mistakes  
As a math idiot  
I want to be told the sum of *two* numbers

Link to a feature: [Calculator](#)

Further read: [Learn more about how to generate Living Documentation](#)

@mytag

### Scenario: Add two numbers

- Given the first number is 50
- And the second number is 70
- When the two numbers are added
- Then the result should be 120



# SpecFlow+ LivingDoc Azure DevOps

Azure DevOps

confoo2022 / Confoo / Overview / SpecFlow+ LivingDoc

Search

Help us to improve LivingDoc in 1-2 minutes! [Give Feedback](#) [Dismiss](#)

## SpecFlow+ LivingDoc

Confoo master generated 13 févr.

Living Documentation Analytics


Filter by Keyword Filter by Scenario Result

Test results

- Confoo2022.SpecFlow.Calculator.Specs
  - Features
    - Calculator
      - Add two numbers
- Confoo2022.SpecFlow.FizzBuzz.Specs
  - Features
    - FizzBuzz
      - Print FizzBuzz

### Feature: Calculator

Open Editor



Simple calculator for adding **two** numbers

Link to a feature: [Calculator](#)

**Further read:** [Learn more about how to generate Living Documentation](#)

@mytag

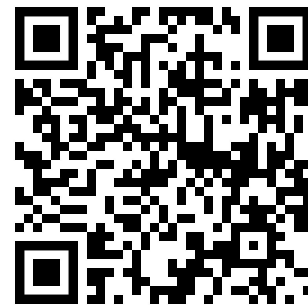
#### Scenario: Add two numbers

- Given the first number is 50
- And the second number is 70
- When the two numbers are added
- Then the result should be 120

# Demo



# specflow



<https://github.com/FrancisGauthier/confoo2022/>

# Conclusion

- Test d'acceptances
  - Avoir une compréhension commune des requis
  - Valide que les exigences sont comblées
  - Vérifie le comportement du système
- Références
  - Site officiel Specflow
    - <https://specflow.org/>
  - SpecFlow BDD Masterclass by Gaspar Nagy
    - <https://specflow.org/school/bdd-masterclass/>

# done.

votre partenaire techno

Découvrez  
[donetechno.com/equipe-gagnante](https://donetechno.com/equipe-gagnante)

MERCI!



[donetechno.com](https://donetechno.com)