

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II



U. PORTO

KU LEUVEN



RISE

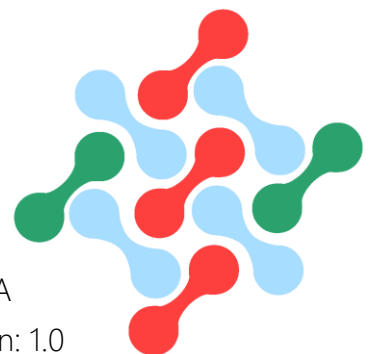
INOVA+
INTERNATIONAL

CESUR
Tu Centro Oficial de FP

CTG
Academy

nexo **qa** ENACTEST

WP4 · Capsule 11



Funded by
the European Union

ERASMUS plus Project 2022-2025

Author: NEXO QA
Document version: 1.0



BBD Test Automation



Prerequisites

- Agile testing concepts
- Java programming knowledge
- IntelliJ IDE
- Maven plugin (IntelliJ)
- Gherkin syntax plugin (IntelliJ)
- Cucumber framework plugin (IntelliJ)



Contents

- BDD theory
- BDD in practice using Cucumber
- Cucumber student exercise



BDD Theory

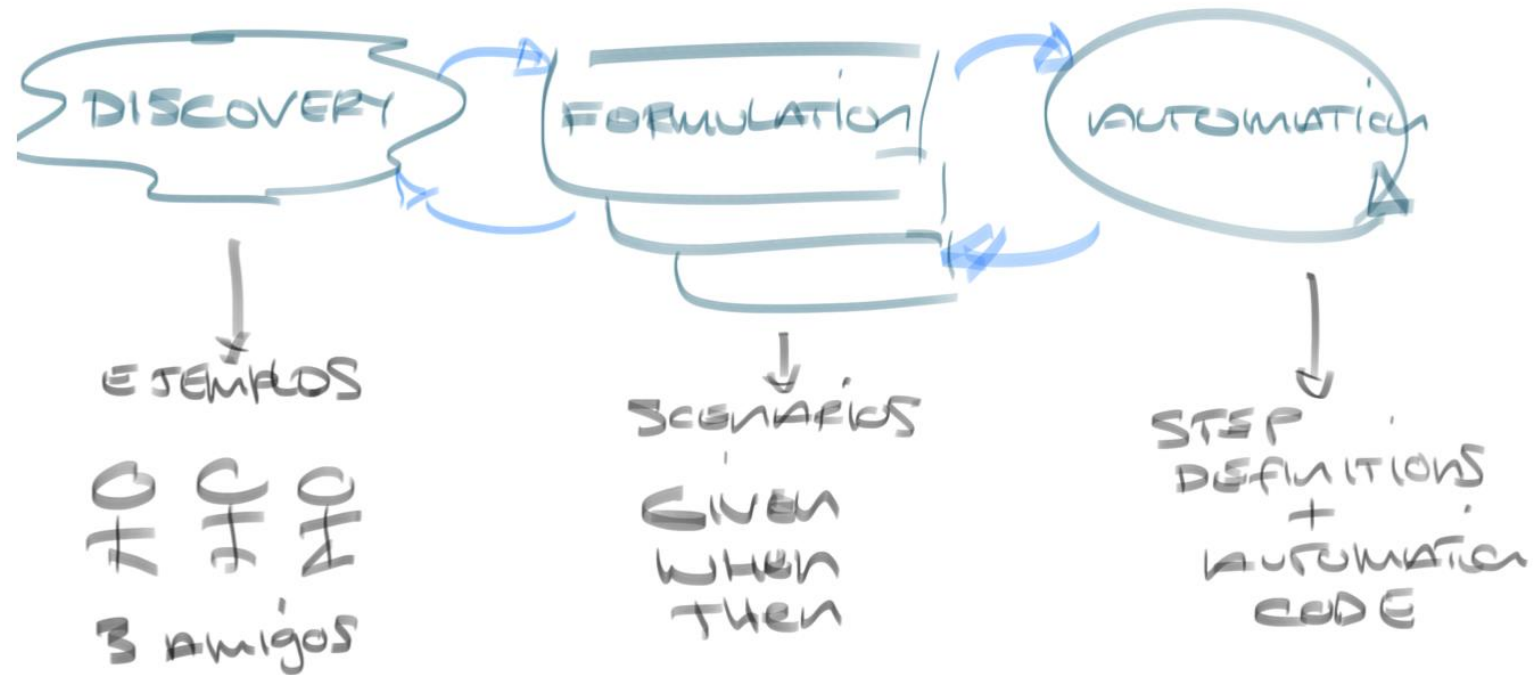


What is BDD?

- A way of working that allows the business to work with the development team
- Business Analysts, Developers, Testers work together to understand the behaviour of software being developed
- Specifications are produced that drive development and are automatically checked against software behaviour ('live' documentation)



BDD Process · Discovery





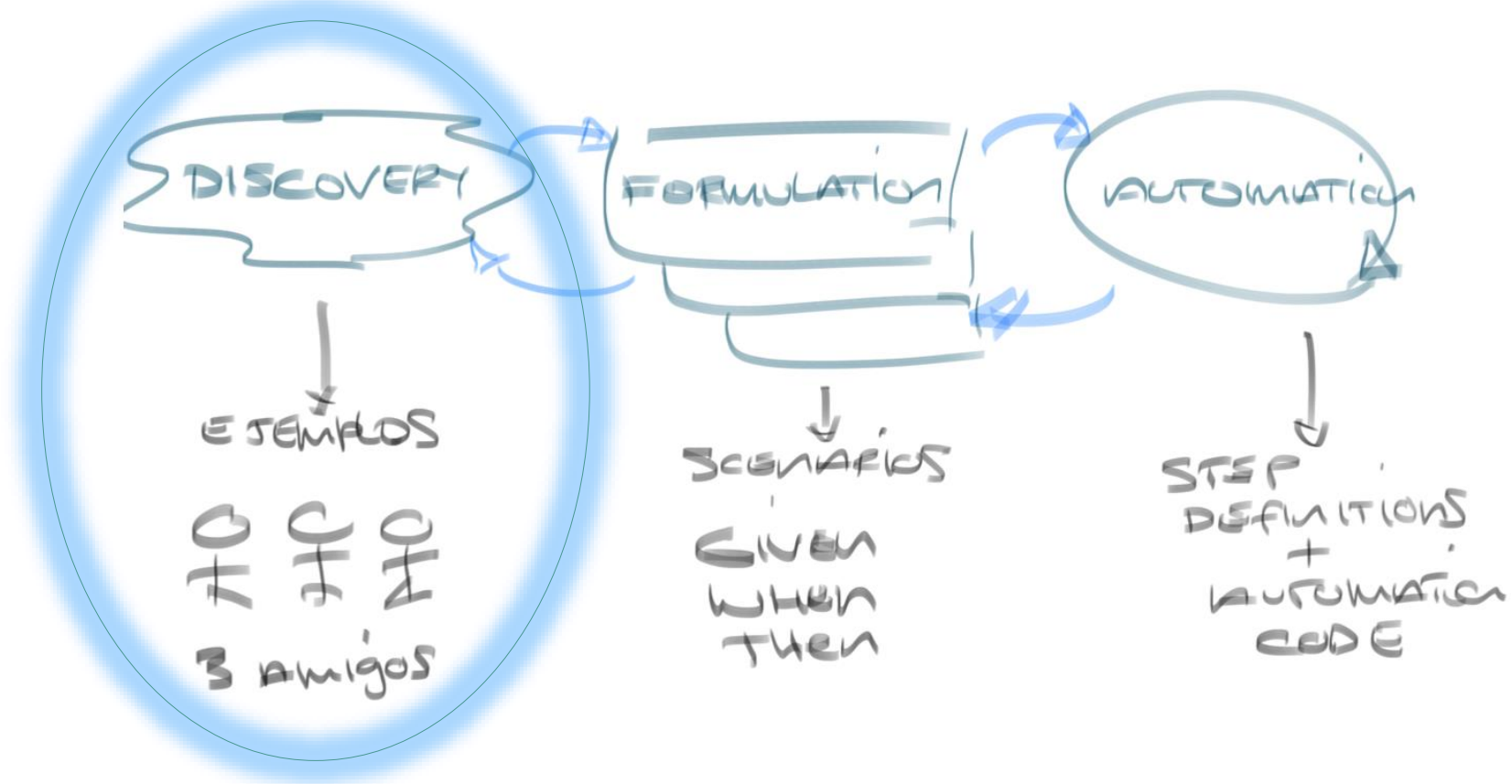
BDD Process

3 practices are repeated constantly:

- 1) Discovery:** User stories are used to explore, discover and agree on how the software should behave by using concrete examples
- 2) Formulation:** These examples are documented using a keyword syntax
- 3) Automation:** Implement the behaviour described by each example using automated tests that drive the development of the software



BDD Process



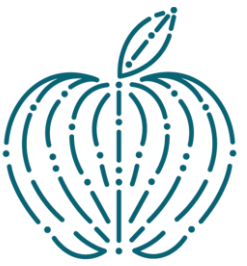


Let's start. . .

An airline company has a customer loyalty club where customers can take advantage of benefits, but the company has noticed a drop in subscriptions. They think it is due to the process of renewing the subscription, where members have to submit an application by email.

A new capability has been identified, 'allowing members to renew their subscription more easily'.

New features have been identified as 'allow members to renew their subscription online' and 'notify members when their subscription is due for renewal'.



Features

"A feature is a functionality that is delivered to end users or other stakeholders to support a capability they need to achieve their business objectives"

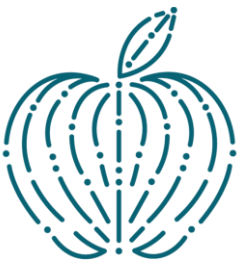
A feature can be expressed using 'for', 'like', 'want':

Feature: Renew online subscription **(the context)**

To renew my subscription more easily **(the business objective)**

As a member of the loyalty club **(the user / stakeholder)**

I want to be able to renew my subscription online **(what you have to do)**



Features

When we break features into parts small enough to build in a single iteration, we can call them user stories:

Renew online subscription (**high level feature**)

Renew using points (**low level feature**)

Renew using credit card (**low level feature**)

Renew using paypal (**low level feature**)

Renew using credit card (**low level feature**)

Pay with visa (**user history**)

Pay with mastercard (**user story**)

Pay with american express (**user story**)



User story

A feature is NOT a user story!!!

"A user story is a planning tool that helps you develop the details of what you need to deliver for a particular feature."



Examples

"An example is used to understand how features will help users and guide the implementation of a user story."

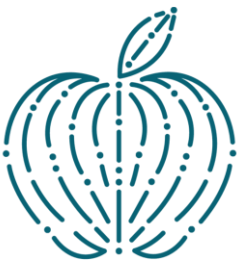
"Examples can be used to understand both features and individual user stories."

"Examples become automated acceptance criteria that developers use as a guide when implementing stories and features."

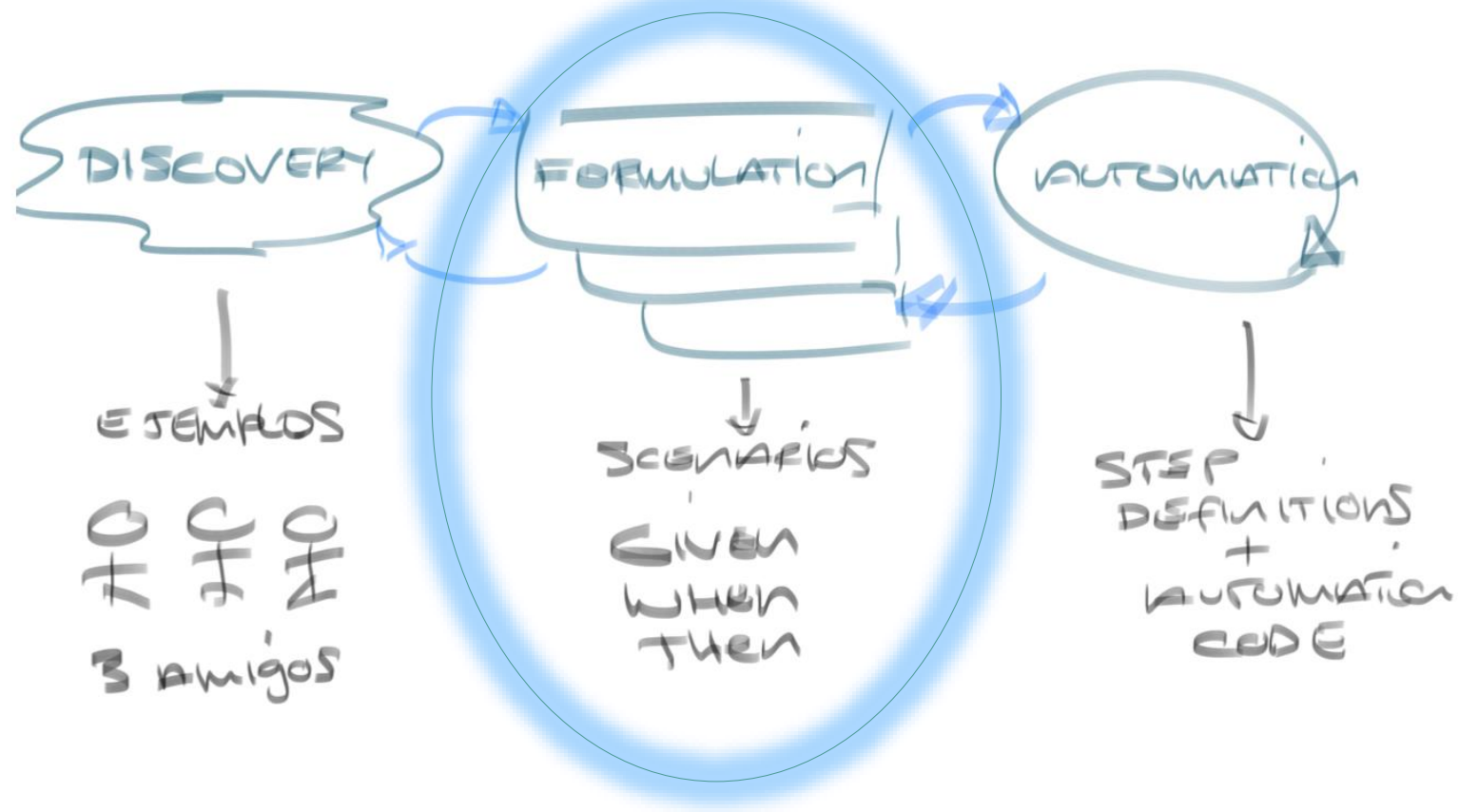


Examples

- We express concrete examples as executable scenarios, using the format of 'Given. . . When. . . Then'.
- We automate this format using Cucumber, a BDD tool.
- We use tables to combine several similar examples more concisely into a single scenario, or to express test data or expected results in a more concise way.
- Scenarios are organised into feature files and can be annotated with tags to indicate cross-functional concerns and coordinate test execution.



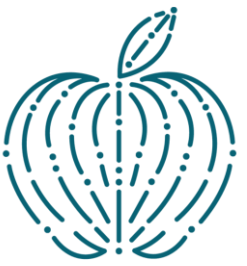
BDD Process · Formulation





Scenarios

"A scenario describes an example of system behaviour and in Cucumber describes an example that can be run."



Scenarios (as Gherkin)

Scenario: **(the example)**

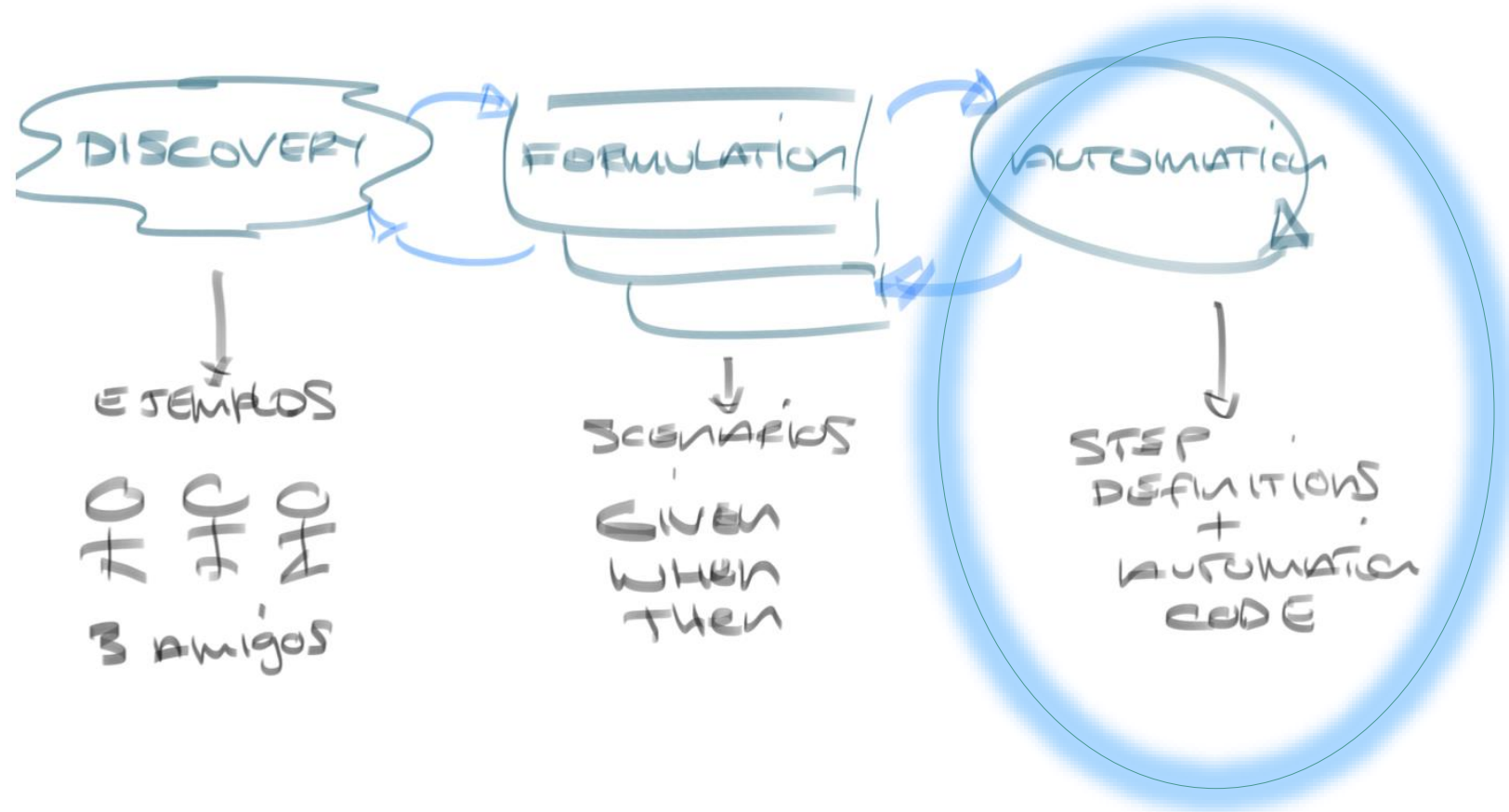
Given **(the context - a concrete state)**

When **(an action - producing a result)**

Then **(the result - the behaviour given the context)**

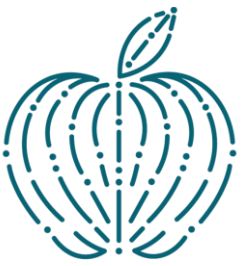


BDD Process · Automation



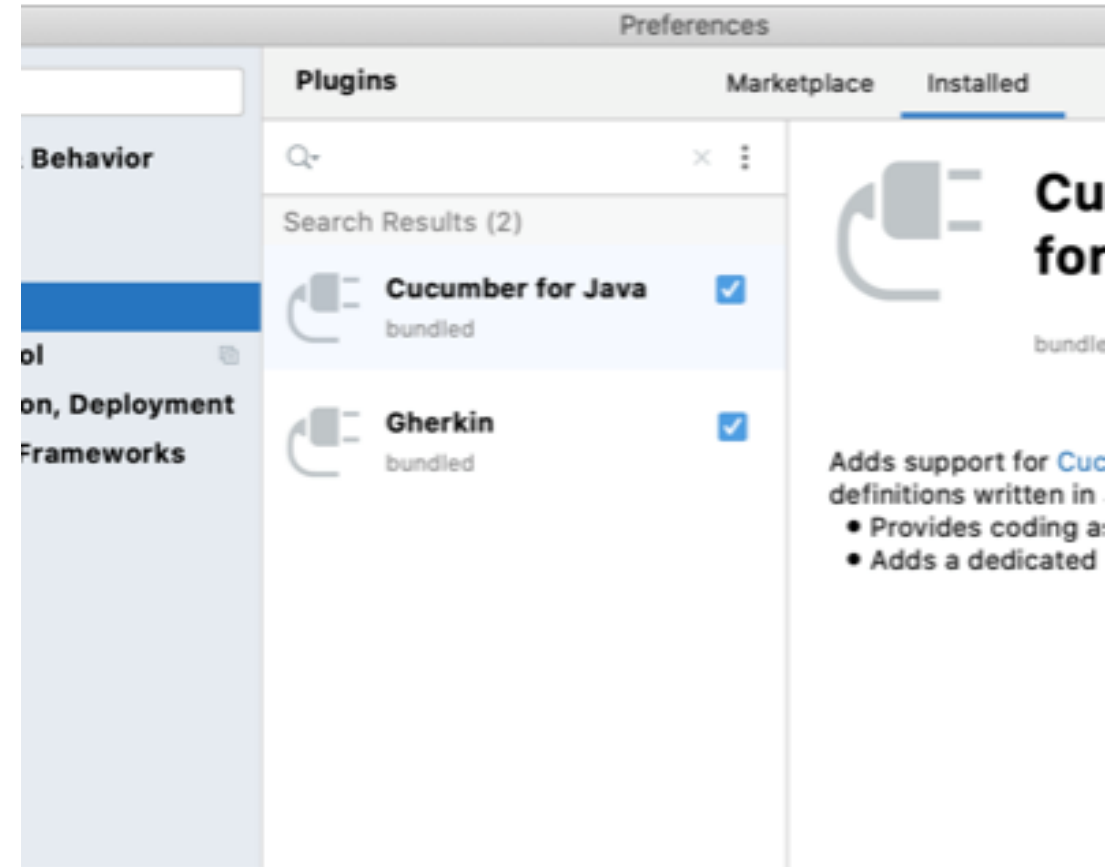


BDD in practice using Cucumber



Let's set things up. . .

- Install IntelliJ IDEA CE
- Clone project on GitHub
- Install Gherkin and Cucumber for Java





Java project

Scenario: **(the example)**

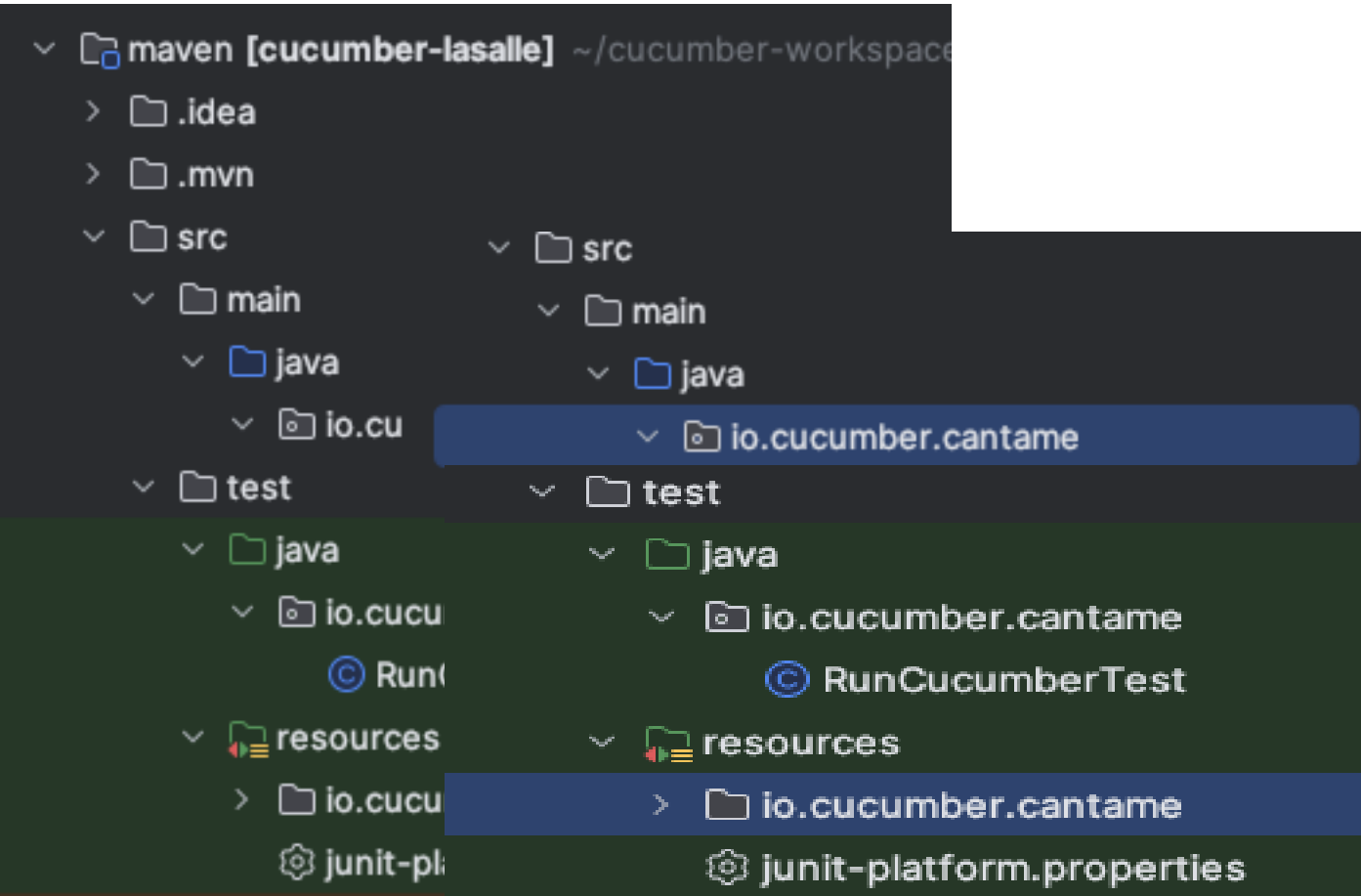
Given **(the context - a concrete state)**

When **(an action - producing a result)**

Then **(the result - the behaviour given the context)**



Java project



Java classes

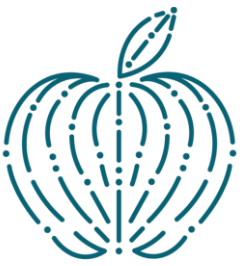
Step definitions

Feature files



My first scenario

- 1) Create feature file
- 2) Define step definitions
- 3) Execute feature file



Create feature file

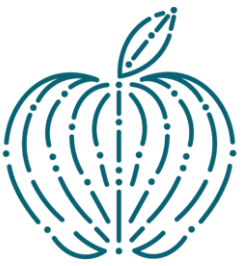
Feature: Escuchar cántame

Scenario: Escuchar dentro del rango

Given Pati está ubicada 10 metros de Javi

When Javi canta café gratuito en mi cafetería

Then Pati escucha al mensaje de Javi



Create feature file

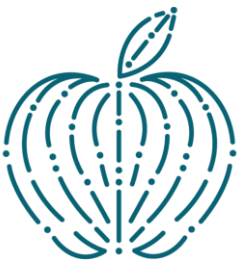
```
1 Scenarios (1 undefined)
3 Steps (2 skipped, 1 undefined)
0m0.751s
```

You can implement missing steps with the snippets below:

```
@Given("Pati está ubicada {int} metros de Javi")
public void pati_está_ubicada_metros_de_javi(Integer int1) {
    // Write code here that turns the phrase above into concrete actions
    throw new io.cucumber.java.PendingException();
}

@When("Javi canta café gratuito en mi cafetería")
public void javi_canta_café_gratuito_en_mi_cafetería() {
    // Write code here that turns the phrase above into concrete actions
    throw new io.cucumber.java.PendingException();
}

@Then("Pati escucha al mensaje de Javi")
public void pati_escucha_al_mensaje_de_javi() {
    // Write code here that turns the phrase above into concrete actions
    throw new io.cucumber.java.PendingException();
}
```



Define step definitions

- Create class **StepDefinitions.java**
- Paste snippets from Cucumber
- Import **given**, **when**, **then** classes
- Run feature file



Define step definitions

```
package io.cucumber.cantame;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class StepDefinitions {

    @Given("Pati está ubicada {int} metros de Javi")
    public void pati_está_ubicada_metros_de_javi(Integer int1) {
        // Write code here that turns the phrase above into concrete actions
        throw new io.cucumber.java.PendingException();
    }

    @When("Javi canta café gratuito en mi cafetería")
    public void javi_canta_café_gratuito_en_mi_cafetería() {
        // Write code here that turns the phrase above into concrete actions
        throw new io.cucumber.java.PendingException();
    }

    @Then("Pati escucha al mensaje de Javi")
    public void pati_escucha_al_mensaje_de_javi() {
        // Write code here that turns the phrase above into concrete actions
        throw new io.cucumber.java.PendingException();
    }
}
```

Run feature file



The screenshot shows the 'Run/Debug Configurations' dialog in an IDE. The configuration is for a Cucumber Java feature file named 'Feature: escuchar_cantame'. The 'Main class' is set to 'io.cucumber.core.cli.Main'. The 'Glue' is set to 'io.cucumber.cantame'. The 'Feature or folder path' is 'pace/cucumber-lasalle/maven/src/test/resources/io/cucumber/cantame/escuchar_cantame.fea'. The 'Program arguments' are '--plugin org.jetbrains.plugins.cucumber.java.run.CucumberJvm5SMFormatter'. The 'Working directory' is '\$MODULE_WORKING_DIR\$'. The 'Environment variables' are 'Environment variables or .env files'. The 'Use classpath of module' is 'cucumber-java'. The 'Shorten command line' is 'none - java [options] className [args]'. The 'Before launch' section is expanded, showing a 'Build' step. The 'Show this page' checkbox is unchecked, 'Activate tool window' is checked, and 'Focus tool window' is unchecked. The 'Run' button is highlighted.

Run/Debug Configurations

Name: Feature: escuchar_cantame Allow multiple instances Store as project file

Configuration Code Coverage Logs

Main class: io.cucumber.core.cli.Main

Glue: io.cucumber.cantame

Feature or folder path: pace/cucumber-lasalle/maven/src/test/resources/io/cucumber/cantame/escuchar_cantame.fea

VM options: + ↕

Program arguments: --plugin org.jetbrains.plugins.cucumber.java.run.CucumberJvm5SMFormatter + ↕

Working directory: \$MODULE_WORKING_DIR\$ + 📁

Environment variables: Environment variables or .env files 📁 📄

Redirect input from: + 📁

Use classpath of module: cucumber-java

JRE: 📁 ▼

Shorten command line: none - java [options] className [args] ▼

Before launch

+ - ✎ ⬆️ ⬇️

🔧 Build

Show this page Activate tool window Focus tool window

Edit configuration templates...

? Run ▼ Cancel Apply OK



Run feature file

```
Step pending
TODO: implement me

Step skipped

Step skipped

Pending scenarios:
file:///Users/graham/cucumber-workspa

1 Scenarios (1 pending)
3 Steps (2 skipped, 1 pending)
0m0.409s

io.cucumber.java.PendingException Crea
at io.cucumber.cantame.StepDefini
at *.Pati está ubicada 10 metros

Process finished with exit code 1

package io.cucumber.cantame;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class StepDefinitions {

    @Given("Pati está ubicada {int} metros de Javi")
    public void pati_está_ubicada_metros_de_javi(Integer int1) {
        // Write code here that turns the phrase above into concrete actions
        throw new io.cucumber.java.PendingException();
    }

    @When("Javi canta café gratuito en mi cafetería")
    public void javi_canta_café_gratuito_en_mi_cafeteria() {
        // Write code here that turns the phrase above into concrete actions
        throw new io.cucumber.java.PendingException();
    }

    @Then("Pati escucha al mensaje de Javi")
    public void pati_escucha_al_mensaje_de_javi() {
        // Write code here that turns the phrase above into concrete actions
        throw new io.cucumber.java.PendingException();
    }
}
```

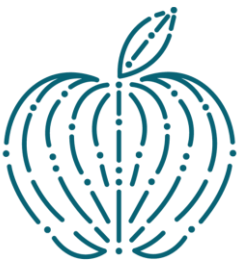


Start adding business logic . . .



Step definition 1 · Business Logic

- Remove exception (pending status)
- Create `Person` object inside step definition (for `pati` and `javi`)
- Create `Person` class
- Call `moveA(distance)` to `pati`
- Create `moveA` method in the `Person` class
- Execute scenario



Step definition 1 · Business Logic

```
package io.cucumber.cantame;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class StepDefinitions {

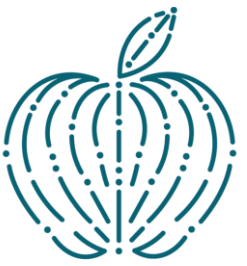
    @Given("Pati está ubicada {int} metros de Javi")
    public void pati_está_ubicada_metros_de_javi(Integer distancia) {
        Persona pati = new Persona();
        Persona javi = new Persona();
        pati.moverA(distancia);
    }

    @When("Javi canta café gratuito en m")
    public void javi_canta_café_gratuito
        // Write code here that turns th
        throw new io.cucumber.java.Pendi
    }

    @Then("Pati escucha al mensaje de Javi")
    public void pati_escucha_al_mensaje_de_javi() {
        // Write code here that turns the phrase above into concrete actions
        throw new io.cucumber.java.PendingException();
    }
}
```

```
package io.cucumber.cantame;

4 usages
public class Persona {
    1 usage
    public void moverA(Integer distancia) {
    }
}
```



Step definition 1 · Run step

```
Step pending
TODO: implement me

Step skipped

Pending scenarios:
file:///Users/graham/cucumber-workspace/cucumber-lasalle/maven/src/test/resources/io/cucumber/cantame/escuchar

1 Scenarios (1 pending)
3 Steps (1 skipped, 1 pending, 1 passed)
0m0.404s

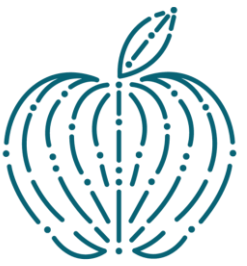
io.cucumber.java.PendingException Create breakpoint : TODO: implement me
  at io.cucumber.cantame.StepDefinitions.javi_canta_café_gratuito_en_mi_cafetería(StepDefinitions.java:19)
  at *.Javi canta café gratuito en mi cafetería(file:///Users/graham/cucumber-workspace/cucumber-lasalle/mav

Process finished with exit code 1
```



Step definition 2 · Business Logic

- Remove exception (pending status)
- Define `javi` as variable (to be accessible between step definitions)
- Call `sing(message)` to `javi`
- Create `sing` method in `Person` class
- Execute scenario



Step definition 2 · Business Logic

```
public class StepDefinitions {  
  
    2 usages  
    private Persona javi;  
  
    2 usages  
    private Persona pati;  
  
    @Given("Pati está ubicada {int} metros de Javi")  
    public void pati_está_ubicada_metros_de_javi(Integer dist  
        pati = new Persona();  
        javi = new Persona();  
        pati.moverA(distancia);  
    }  
  
    @When("Javi canta {string}")  
    public void javi_canta_café_gratuito_en_mi_cafetería(Stri  
        javi.canta(mensaje);  
    }  
  
    @Then("Pati escucha al mensaje de Javi")  
    public void pati_escucha_al_mensaje_de_javi() {  
        // Write code here that turns the phrase above into concrete actions  
        throw new io.cucumber.java.PendingException();  
    }  
}
```

```
package io.cucumber.cantame;  
  
4 usages  
public class Persona {  
    1 usage  
    public void moverA(Integer distancia) {  
    }  
  
    1 usage  
    public void canta(String mensaje) {  
    }  
}
```



Step definition 2 · Run step

```
Step pending
TODO: implement me

Pending scenarios:
file:///Users/graham/cucumber-workspace/cucumber-lasalle/maven/src/test/resources/io/cucumber/cantame/e

1 Scenarios (1 pending)
3 Steps (1 pending, 2 passed)
0m0.418s

io.cucumber.java.PendingException Create breakpoint : TODO: implement me
  at io.cucumber.cantame.StepDefinitions.pati_escucha_al_mensaje_de_javi(StepDefinitions.java:27)
  at *.Pati escucha al mensaje de Javi(file:///Users/graham/cucumber-workspace/cucumber-lasalle/maven

Process finished with exit code 1
```



Step definition 3 · Business Logic

- Remove exception (pending status)
- Create a check (`assertEquals`)
- Import `assertEquals` method from Junit and `asList` method
- Create a variable to store Javi's message
- Create a `getMessageSung` method for pati to receive the message
- Execute scenario



Step definition 3 · Business Logic

```
package io.cucumber.cantame;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

import static java.util.Arrays.asList;
import static org.junit.jupiter.api.Assertions.assertEquals;

public class StepDefinitions {

    2 usages
    private Persona javi;
    3 usages
    private Persona pati;
    2 usages
    private String mensajeDeJavi;

    @Given("Pati está ubicada {int} metros de Javi")
    public void pati_está_ubicada_metros_de_javi(Integer distancia) {
        pati = new Persona();
        javi = new Persona();
        pati.moverA(distancia);
    }

    @When("Javi canta {string}")
    public void javi_canta_café_gratuito_en_mi_cafetería(String mensaje) {
        javi.canta(mensaje);
        mensajeDeJavi = mensaje;
    }

    @Then("Pati escucha al mensaje de Javi")
    public void pati_escucha_al_mensaje_de_javi() {
        assertEquals(asList(mensajeDeJavi), pati.getMensajeCantado());
    }
}
```

```
package io.cucumber.cantame;

import java.util.List;

4 usages
public class Persona {

    1 usage
    public void moverA(Integer distancia) {
    }

    1 usage
    public void canta(String mensaje) {
    }

    1 usage
    public List<String> getMensajeCantado() {
        return null;
    }
}
```



Step definition 3 · Run step

```
Step failed
Expected :[café gratuito en mi cafetería]
Actual   :null
<Click to see difference>

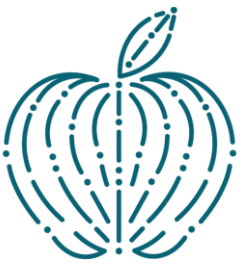
org.opentest4j.AssertionFailedError: Create breakpoint : expected: <[café gratuito en mi cafetería]> but
    at io.cucumber.cantame.StepDefinitions.pati_escucha_al_mensaje_de_javi(StepDefinitions.java:31)
    at *.Pati escucha al mensaje de Javi(file:///Users/graham/cucumber-workspace/cucumber-lasalle/ma

Failed scenarios:
file:///Users/graham/cucumber-workspace/cucumber-lasalle/maven/src/test/resources/io/cucumber/cantam

1 Scenarios (1 failed)
3 Steps (1 failed, 2 passed)
0m0.356s

org.opentest4j.AssertionFailedError: Create breakpoint : expected: <[café gratuito en mi cafetería]> but
    at io.cucumber.cantame.StepDefinitions.pati_escucha_al_mensaje_de_javi(StepDefinitions.java:31)
    at *.Pati escucha al mensaje de Javi(file:///Users/graham/cucumber-workspace/cucumber-lasalle/ma

Process finished with exit code 1
```



Step definition 3 · Fix & run again!

```
import java.util.List;

4 usages
public class Persona {

    1 usage
    public void moverA(Integer distancia) {
    }

    1 usage
    public void canta(String mensaje) {
    }

    1 usage
    public List<String> getMensajeCantado() {
        List<String> result = new ArrayList <String>();
        result.add("café gratuito en mi cafetería");
        return result;
    }
}
```

```
Testing started at 17:54 ...
café gratuito en mi cafetería
```

```
1 Scenarios (1 passed)
3 Steps (3 passed)
0m0.442s
```

```
Process finished with exit code 0
```



Let's look at Cucumber parameters . . .



Cucumber Expressions · Parameters

Feature: Escuchar cántame

Scenario: Escuchar dentro del rango

Given Pati está ubicada **10** metros de Javi

When Javi canta café gratuito en mi cafetería

Then Pati escucha al mensaje de Javi

```
@Given("Pati está ubicada {int} metros de Javi")
```

```
public void pati_está_ubicada_metros_de_javi(Integer distancia) {
```

```
    pati = new Persona();
```

```
    javi = new Persona();
```

```
    pati.moverA(distancia);
```

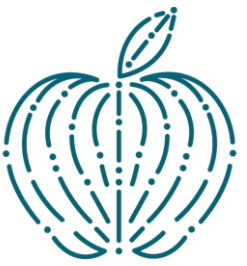
```
}
```

```
Pati está 1000 centímetros de Javi  
café gratuito en mi cafetería
```

```
1 Scenarios (1 passed)
```

```
3 Steps (3 passed)
```

```
0m0.398s
```



Cucumber Expressions · Parameters

Feature: Escuchar cántame

Scenario: Escuchar dentro del rango

Given Pati está ubicada **1 metro** de Javi

When Javi canta café gratuito en mi cafetería

Then Pati escucha al mensaje de Javi

```
Pati está 100 centímetros de Javi
café gratuito en mi cafetería
```

```
@Given("Pati está ubicada {int} metros de Javi")
public void pati_está_ubicada_metros_de_javi(Integer distancia) {
    pati = new Persona();
    javi = new Persona();
    pati.moverA(distancia);
    System.out.println(String.format("Pati está %d centímetros de Javi", distancia * 100));
}
```

```
1 Scenarios (1 passed)
3 Steps (3 passed)
0m0.392s
```



Cucumber Expressions · Parameters

Feature: Escuchar cántame

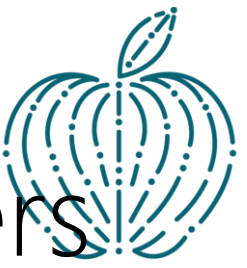
Scenario: Escuchar dentro del rango

Given Pati está localizada 1 metro de Javi

When Javi canta café gratuito en mi cafetería

Then Pati escucha al mensaje de Javi

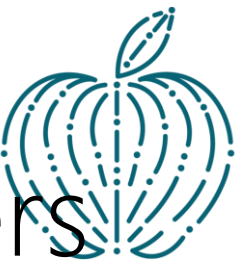
```
@Given("Pati está ubicada/localizada {int} metro(s) de Javi")
public void pati_está_ubicada_metro_de_javi(Integer distancia) {
    pati = new Persona();
    javi = new Persona();
    pati.moverA(distancia);
    System.out.println(String.format("Pati está %d centímetros de Javi", distancia * 100));
}
```



Cucumber Expressions · Custom parameters

Custom parameters help to further link business logic with scenarios

- Create person parameter
 - Replace 'pati' with parameter {person}
 - Create a 'support' package in 'test.java.io.cucumber.cantame'.
 - Create the class 'ParameterTypes'.
 - Create the 'Person' method
 - Create the method 'Persona' in the class Persona
 - Import the class Persona and ParameterType



Cucumber Expressions · Custom parameters

```
@Given("{persona} está ubicada/localizada {int} metro(s) de Javi")
public void pati_está_ubicada_metro_de_javi(Persona persona, Integer distancia) {
    pati = new Persona();
    javi = new Persona();
    pati.moverA(distancia);
    sy package io.cucumber.cantame.support;           de Javi", distancia * 100));
}
```

```
import io.cucumber.java.ParameterType;
import io.cucumber.cantame;

package io.cucumber.cantame;

import java.util.ArrayList;
import java.util.List;

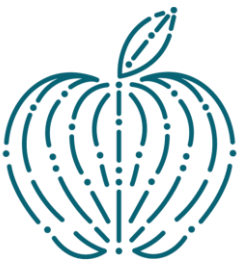
public class ParameterType {
    @ParameterType("Pat")
    public Persona pers
        return new Pers
    }
}

public class Persona {

    public Persona(String nombre) {
    }
}
```



Let's look at data tables . . .



Cucumber data tables

Feature: Escuchar cántame

Scenario: Escuchar dentro del rango

Given <nombre> está ubicada 10 metros de <nombre>

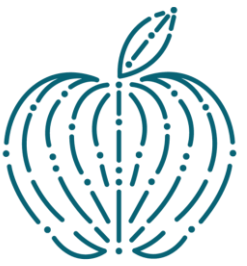
| nombre |

| Pati |

| Javi |

When Javi canta café gratuito en mi cafetería

Then Pati escucha al mensaje de Javi



Cucumber data tables

```
@Given("<nombre> está ubicada/localizada {int} metros de <nombre>")
public void nombre_está_ubicada_metros_de_nombre(Integer distancia, DataTable dataTable) {
    cliente = new Persona(dataTable.cell(row: 1, column: 0));
    dueño = new Persona(dataTable.cell(row: 2, column: 0));
    cliente.moverA(distancia);
}

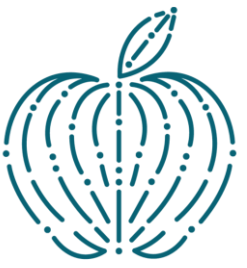
@Given("Javi está ubicado {int} metro de su cafetería")
public void javi_está_ubicado_metro_de_su_cafetería(Integer distancia) {
    dueño.moverA(distancia);
}

@When("Javi canta {string}")
public void javi_canta_café_gratuito_en_mi_cafetería(String mensaje) {
    dueño.canta(mensaje);
    mensajeDeDueño = mensaje;
}

@Then("Pati escucha al mensaje de Javi")
public void pati_escucha_al_mensaje_de_javi() {
    assertEquals(asList(mensajeDeDueño), cliente.getMensajeCantado());
}
```



Try it for yourself



Cucumber student exercise

Define BDD scenarios using Gherkin (Given-When-Then) to test the www.vueling.com login.

The project should be delivered with:

- Feature file
- Step definitions
- Login class (can be empty methods)



Next steps . . .



To do . . .

- Develop teacher manual & videos
- Add a few more aspects of BDD
- Add cucumber unit testing
- Add serenity front-end testing using BDD (may be too long though)