

# Acoplamiento de Cucumber (lenguaje Gherkin)

Manotoa Alava Carlos Alexander

[GitHub](#)

## ¿Qué es Cucumber?

Es una herramienta diseñada para ejecutar pruebas de aceptación automatizada expresadas en lenguaje natural; es decir, permite expresar el comportamiento esperado de un sistema con palabras que cualquier miembro del equipo pueda entender, mejorando así la comunicación y colaboración [1].

## ¿Qué es Gherkin?

Es un lenguaje específico del dominio (DSL) que puede entenderse tanto por perfiles de negocios como técnicos e incluso ser traducido en código de manera automática, consiste en una estructura de elementos y características que describen el comportamiento y acciones de manera sencilla [2].

Gherkin se basa en una sintaxis específica, con un lenguaje natural y fácil de seguir. Un escenario comienza con la palabra clave “Given (Dado)” que define el contexto inicial, “When (Cuando)” procede un acontecimiento, evento o acción específica, y por último “Then (Entonces)” un resultado determinado u una secuencia prevista [3].

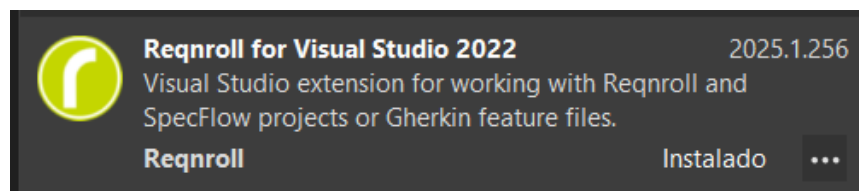
## INVESTIGAR EL ACOPLAMIENTO DE CUCUMBER (LENGUAJE GHERKIN) CON LA PLATAFORMA DE C#

Para poder conectar C# con Cucumber, usaremos la herramienta ReqnRoll que es la versión .NET de Cucumber basada en el framework y código de SpecFlow [4].

### • INSTALACIÓN

Para utilizar ReqnRoll con Visual Studio 2022, debe instalar la extensión [Reqnroll para Visual Studio 2022](#) [5].

1. Abrir Visual Studio 2022
2. En *Extensiones*, elija el *Administrar extensiones*
3. Asegúrese de estar en la opción de *Examinación* y escriba “ReqnRoll” que se muestra en la *imagen 1*



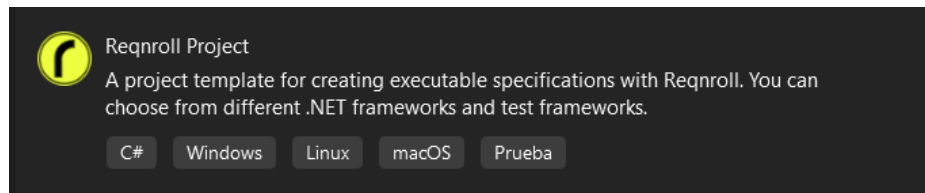
**Tabla 1:** Extensión [Reqnroll para Visual Studio 2022](#)

4. Seleccione *ReqnRoll for Visual Studio 2022* y haga clic en el botón *Descargar*
5. Reinicie Visual Studio 2022

- **ESCRITURA DE ESCENARIOS**

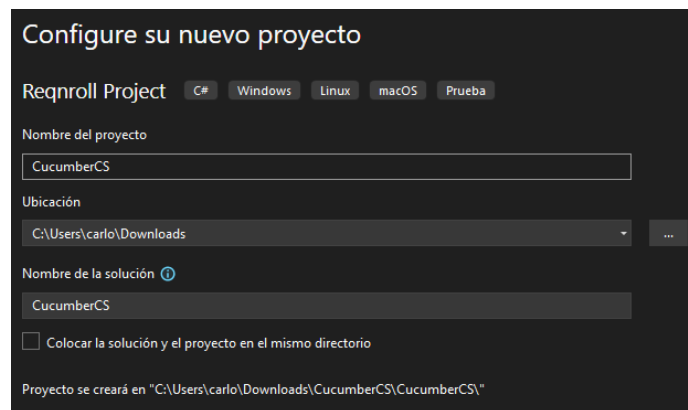
A continuación, se redactará como escribir escenarios en C#. Estos pasos son guiados por videos del canal de YouTube [Execute Automation](#).

1. Abrir Visual Studio 2022 y crear un nuevo proyecto del tipo *Reqnroll Project* que se muestra en la *imagen 2*



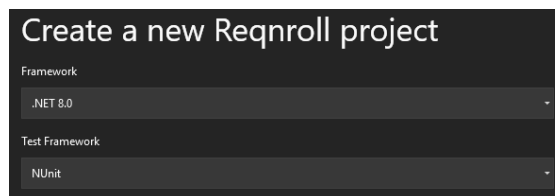
**Imagen 2:** Plantilla de proyecto Reqnroll Project

2. Dar un nombre al proyecto como se muestra en la *imagen 3*



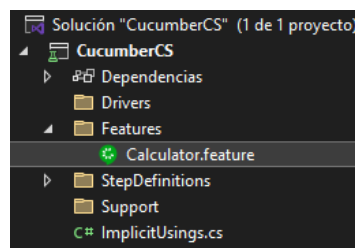
**Imagen 3:** Dar un nombre al proyecto

3. Seleccionar el Framework *.NET 8.0* (O superior) y el Test Framework *NUnit* como se muestra en la imagen 4



**Imagen 4:** Seleccionar Framework y Test Framework

**Nota:** El usar la plantilla nos genera una estructura de carpetas y agrega las dependencias de manera automática, incluyendo un ejemplo llamado “Calculadora.feature”. Es en la carpeta *Features* donde se redactan los escenarios.



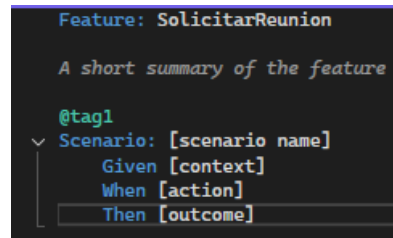
**Imagen 5:** Estructura de carpetas creada automáticamente

4. Para agregar un escenario dar clic derecho sobre la carpeta llamada *Features* → *Agregar* → *Nuevo elemento*. Seleccionar la opción *Feature File for Reqnroll* y darle un nombre.



**Imagen 6:** Elemento *Feature File for Reqnroll*

5. El elemento *Feature File for Reqnroll* crea una plantilla para el nuevo escenario.



**Imagen 7:** Plantilla generada automáticamente

Los escenarios deben tener una estructura mínima obligatoria:

Feature: <nombre de la funcionalidad>

@tag

Scenario: <título del caso>

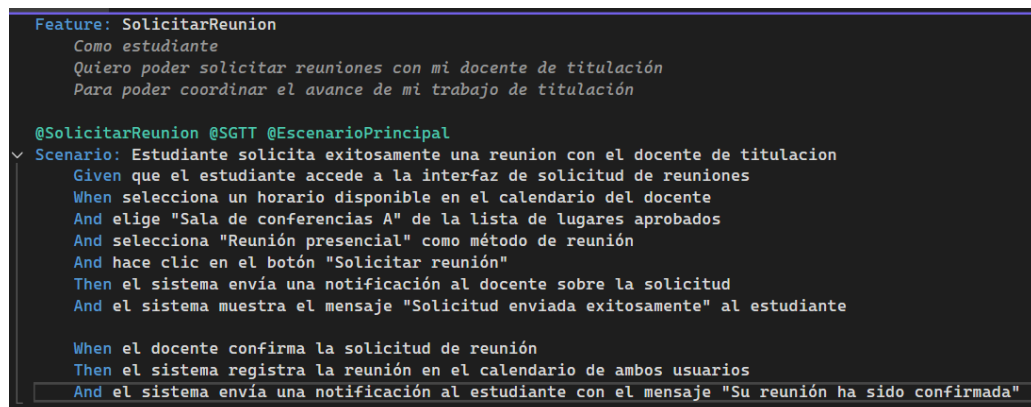
Given <contexto inicial>

When<Acción o evento>

Then<Resultados>

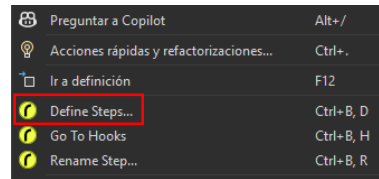
Opcional: And / But para concatenar pasos sin repetir palabras clave.

A continuación, en la *imagen 8* se muestra el escenario para la función “Solicitar Reunión” del sistema SGT, en base a una historia de usuario.



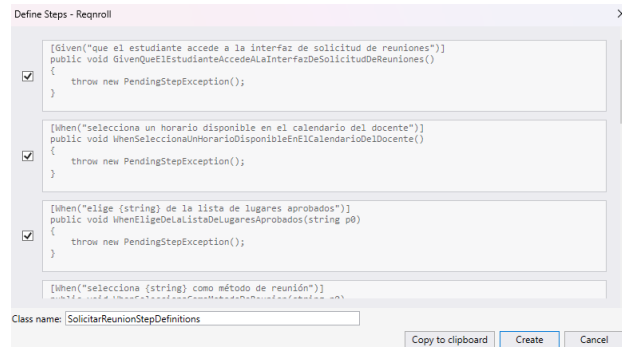
**Imagen 8:** Redacción del escenario para la funcionalidad “Solicitar Reunión”

6. Compilar el proyecto y sobre una de las palabras reservadas dar clic derecho y seleccionar *Define Steps*



**Imagen 9:** Seleccione *Define Steps*

7. Seleccionar todo y dar clic sobre el botón *Create*



**Imagen 10:** Crear definición

Esto creará un archivo .cs en la carpeta *StepDefinitions* que contendrá un esqueleto con todos los pasos definidos.

- **PRUEBA Y VALIDACIÓN**

1. Para reducir la cantidad de pulsaciones de teclas cambiar las *definiciones de paso pendiente* por `Console.WriteLine("Paso")` para que el copiloto lo genere de manera automática.

```
[Then("el sistema muestra el mensaje {string} al estudiante")]
0 referencias
public void ThenElSistemaMuestraElMensajeAlEstudiante(string p0)
{
    throw new PendingStepException();
}
```

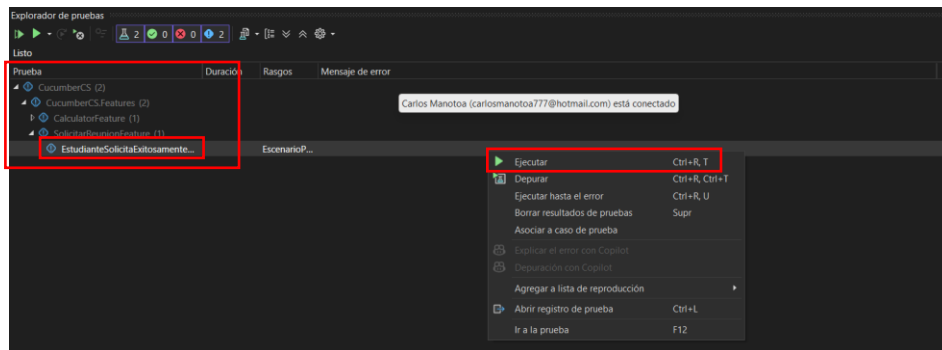
**Imagen 11:** Código con definición de paso pendiente

```
[Then("el sistema muestra el mensaje {string} al estudiante")]
0 referencias
public void ThenElSistemaMuestraElMensajeAlEstudiante(string p0)
{
    Console.WriteLine("El sistema muestra el mensaje " + p0 + " al estudiante");
}
```

**Imagen 12:** Definición de paso pendiente reemplazado por `Console.WriteLine("")`

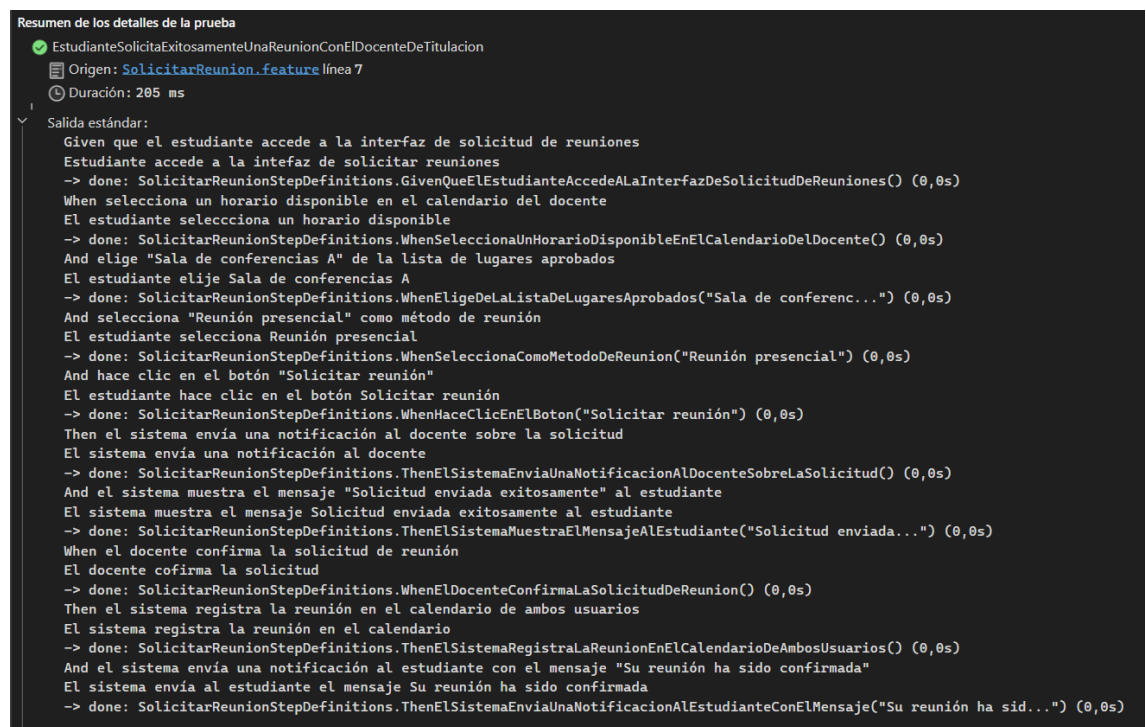
- Después de reemplazar todas las definiciones de paso pendiente, dar clic sobre *prueba* → *Explorador de pruebas*

Desde aquí se pueden ejecutar los distintos escenarios existentes.



*Imagen 13: Explorador de pruebas*

Una vez ejecutado el escenario se muestra la ejecución con sus detalles



**Imagen 14: Ejecución del escenario**

“EstudianteSolicitaExitosamenteUnaReunionConElDocenteDeTitulacion”

### Videos de referencia:

<https://youtu.be/UQDQZ1aANdU?si=gu7NiUpvJW9rCF4v>

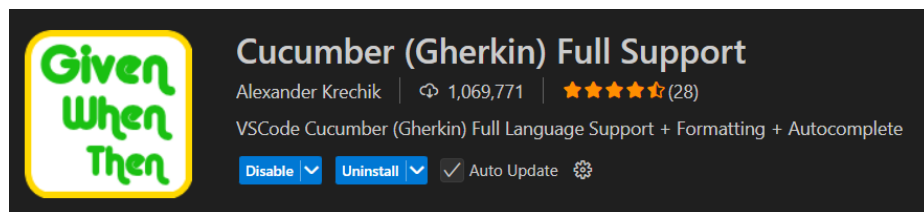
[https://youtu.be/c06ao\\_jf6R4?si=xq0LWkllrpWT7JAK](https://youtu.be/c06ao_jf6R4?si=xq0LWkllrpWT7JAK)

## DEMOSTRAR LA AUTOMATIZACIÓN DE PRUEBAS EN CUCUMBER + PYTHON

Para integrar Cucumber en un proyecto de Python, se puede utilizar implementaciones como behave o pytest-bdd [6], [7].

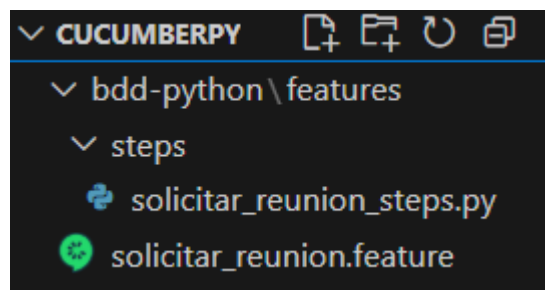
- **Instalación**

1. Instalar [Python](#)
2. Verificar que Python se instaló correctamente con el comando `Python --version`
3. Instalar *behave* usando el comando `pip install behave`
4. Verificar si se instaló correctamente *behave* usando el comando `behave --version`
  - Instalar la extensión *Cucumber (Gherkin) Full Support* – *alexkrechik.cucumberautocomplete*



**Imagen 15:** Extensión *Cucumber (Gherkin) Full Support*

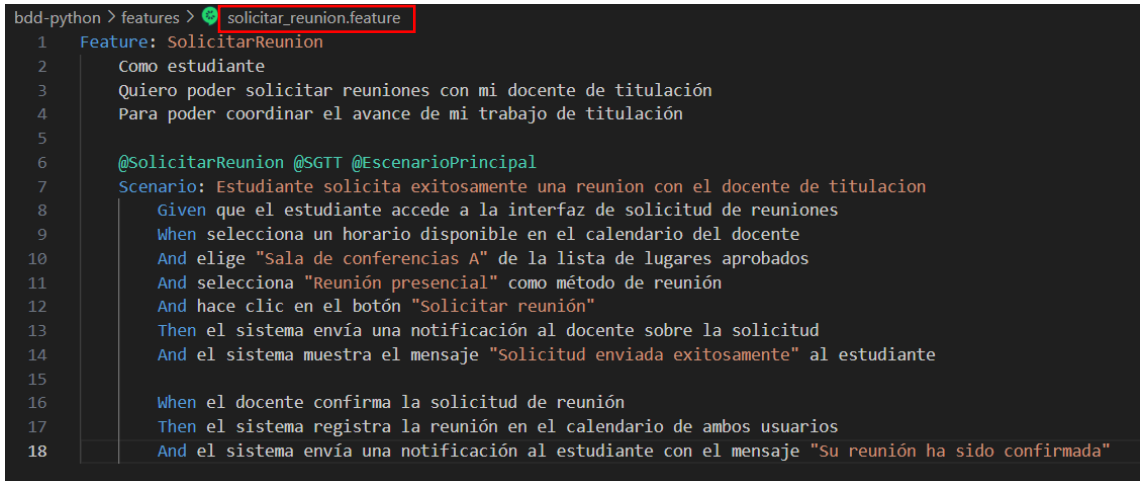
5. Crear la carpeta raíz e ingresar
  - Crear la carpeta raíz con el comando `mkdir <nombre>`  
**Ejemplo:** `mkdir bdd-python`
  - Ingresar dentro de la carpeta raíz usando `cd <nombre>`  
**Ejemplo:** `cd bdd-python`
6. Dentro del proyecto crear una carpeta llamada “features” donde se redactarán los escenarios con la extensión `.feature`
7. Crear una carpeta llamada “Steps” donde se definirá los pasos `.py`
8. Da Dentro de la carpeta “features” crear un archivo. Features con el nombre del escenario



**Imagen 16:** Estructura del programa

- **Escritura de escenarios**

1. Dentro de la carpeta “features” crear un archivo .feature y redactar el escenario

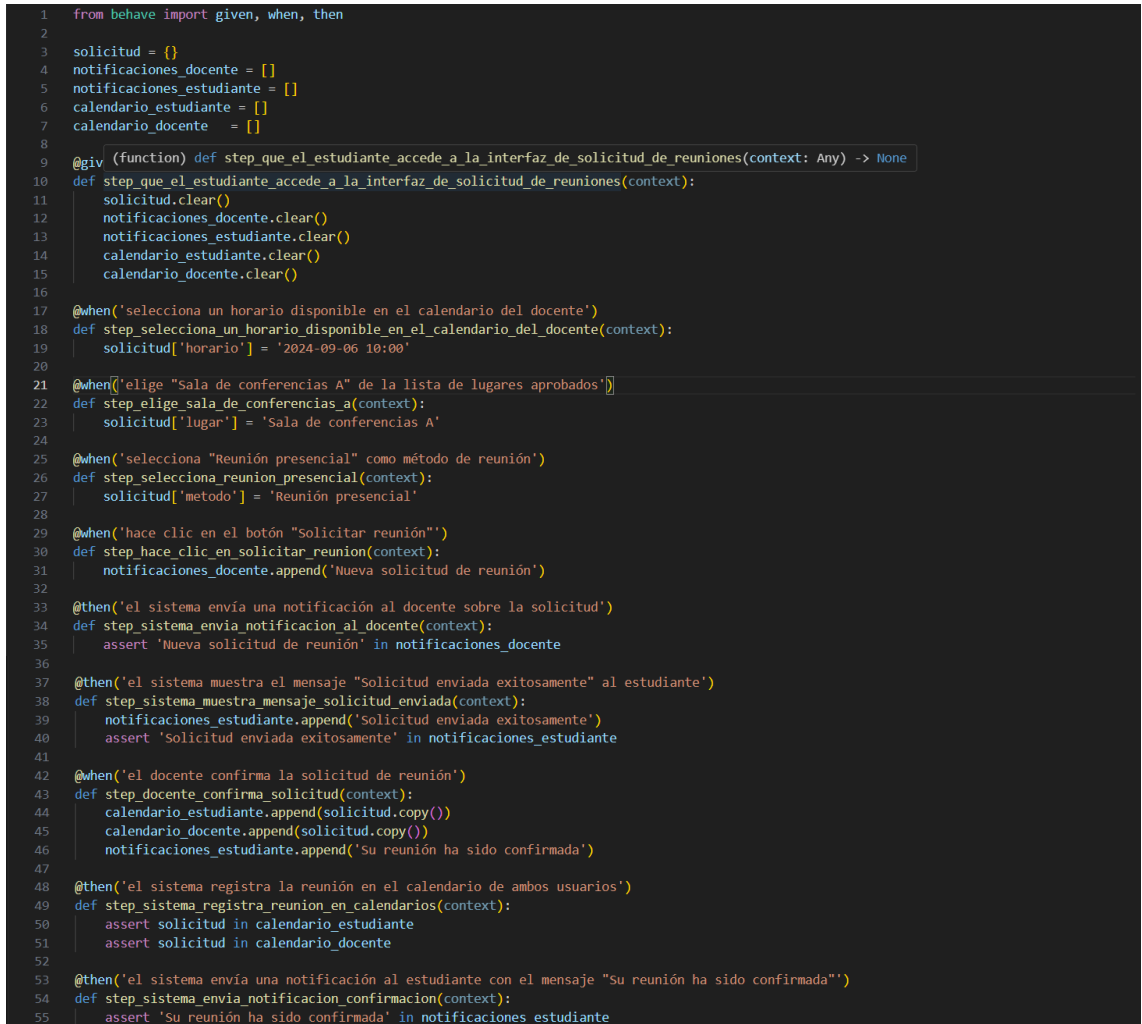


```

bdd-python > features > solicitar_reunion.feature
1 Feature: SolicitarReunion
2   Como estudiante
3   Quiero poder solicitar reuniones con mi docente de titulación
4   Para poder coordinar el avance de mi trabajo de titulación
5
6   @SolicitarReunion @SGTT @EscenarioPrincipal
7   Scenario: Estudiante solicita exitosamente una reunion con el docente de titulacion
8     Given que el estudiante accede a la interfaz de solicitud de reuniones
9     When selecciona un horario disponible en el calendario del docente
10    And elige "Sala de conferencias A" de la lista de lugares aprobados
11    And selecciona "Reunión presencial" como método de reunión
12    And hace clic en el botón "Solicitar reunión"
13    Then el sistema envía una notificación al docente sobre la solicitud
14    And el sistema muestra el mensaje "Solicitud enviada exitosamente" al estudiante
15
16    When el docente confirma la solicitud de reunión
17    Then el sistema registra la reunión en el calendario de ambos usuarios
18    And el sistema envía una notificación al estudiante con el mensaje "Su reunión ha sido confirmada"
  
```

*Imagen 17: Descripción del escenario en Visual Studio Code*

2. En la carpeta “Steps” crear un archivo .py donde se definirán los pasos



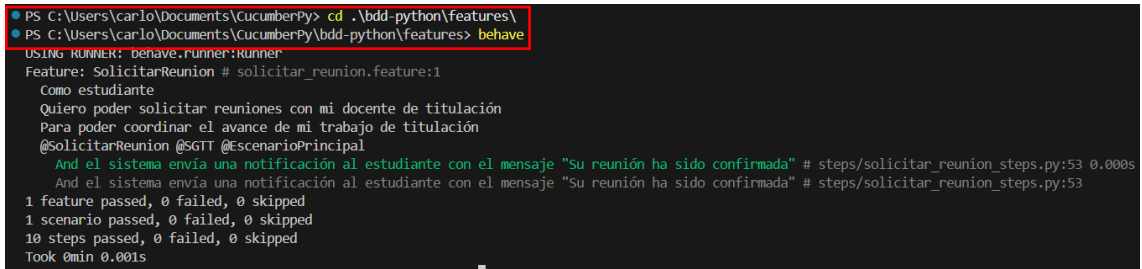
```

1 from behave import given, when, then
2
3 solicitud = {}
4 notificaciones_docente = []
5 notificaciones_estudiante = []
6 calendario_estudiante = []
7 calendario_docente = []
8
9 @given('selecciona un horario disponible en el calendario del docente')
10 def step_que_el_estudiante_accede_a_la_interfaz_de_solicitud_de_reuniones(context):
11     solicitud.clear()
12     notificaciones_docente.clear()
13     notificaciones_estudiante.clear()
14     calendario_estudiante.clear()
15     calendario_docente.clear()
16
17 @when('selecciona un horario disponible en el calendario del docente')
18 def step_selecciona_un_horario_disponible_en_el_calendario_del_docente(context):
19     solicitud['horario'] = '2024-09-06 10:00'
20
21 @when('elige "Sala de conferencias A" de la lista de lugares aprobados')
22 def step_elige_sala_de_conferencias_a(context):
23     solicitud['lugar'] = 'Sala de conferencias A'
24
25 @when('selecciona "Reunión presencial" como método de reunión')
26 def step_selecciona_reunion_presencial(context):
27     solicitud['metodo'] = 'Reunión presencial'
28
29 @when('hace clic en el botón "Solicitar reunión"')
30 def step_hace_clic_en_solicitar_reunion(context):
31     notificaciones_docente.append('Nueva solicitud de reunión')
32
33 @then('el sistema envía una notificación al docente sobre la solicitud')
34 def step_sistema_envia_notificacion_al_docente(context):
35     assert 'Nueva solicitud de reunión' in notificaciones_docente
36
37 @then('el sistema muestra el mensaje "Solicitud enviada exitosamente" al estudiante')
38 def step_sistema_muestra_mensaje_solicitud_enviada(context):
39     notificaciones_estudiante.append('Solicitud enviada exitosamente')
40     assert 'Solicitud enviada exitosamente' in notificaciones_estudiante
41
42 @when('el docente confirma la solicitud de reunión')
43 def step_docente_confirma_solicitud(context):
44     calendario_estudiante.append(solicitud.copy())
45     calendario_docente.append(solicitud.copy())
46     notificaciones_estudiante.append('Su reunión ha sido confirmada')
47
48 @then('el sistema registra la reunión en el calendario de ambos usuarios')
49 def step_sistema_registra_reunion_en_calendarios(context):
50     assert solicitud in calendario_estudiante
51     assert solicitud in calendario_docente
52
53 @then('el sistema envía una notificación al estudiante con el mensaje "Su reunión ha sido confirmada"')
54 def step_sistema_envia_notificacion_confirmacion(context):
55     assert 'Su reunión ha sido confirmada' in notificaciones_estudiante
  
```

*Imagen 18: Definición de los Steps en Visual Studio Code*

- **Prueba y validación**

Una vez estén listos los archivos .feature y .py, ejecutar *behave* desde la carpeta “Features”



```
PS C:\Users\carlo\Documents\CucumberPy> cd .\bdd-python\features\  
PS C:\Users\carlo\Documents\CucumberPy\bdd-python\features> behave  
USING RUNNER: behave.runner:runner  
Feature: SolicitarReunion # solicitar_reunion.feature:1  
  Como estudiante  
  Quiero poder solicitar reuniones con mi docente de titulación  
  Para poder coordinar el avance de mi trabajo de titulación  
  @SolicitarReunion @SGTT @EscenarioPrincipal  
    And el sistema envía una notificación al estudiante con el mensaje "Su reunión ha sido confirmada" # steps/solicitar_reunion_steps.py:53 0.000s  
    And el sistema envía una notificación al estudiante con el mensaje "Su reunión ha sido confirmada" # steps/solicitar_reunion_steps.py:53  
1 feature passed, 0 failed, 0 skipped  
1 scenario passed, 0 failed, 0 skipped  
10 steps passed, 0 failed, 0 skipped  
Took 0min 0.001s
```

*Imagen 19: Ejecución en Visual Studio Code*

## Referencias

- [1] “Pepino.” Accessed: Aug. 29, 2025. [Online]. Available: <https://cucumber.io/>
- [2] “BDD y Gherkin para comunicar mundos diferentes - Paradigma.” Accessed: Aug. 29, 2025. [Online]. Available: <https://www.paradigmadigital.com/dev/bdd-gherkin-comunicar-mundos-diferentes/>
- [3] “Gherkin para escribir historias de usuario.” Accessed: Aug. 29, 2025. [Online]. Available: <https://www.media.thiga.co/es/gherkin>
- [4] “Documentación de Reqnroll.” Accessed: Aug. 29, 2025. [Online]. Available: <https://docs.reqnroll.net/latest/>
- [5] “Configurar un IDE para Reqnroll - Documentación de Reqnroll.” Accessed: Aug. 29, 2025. [Online]. Available: <https://docs.reqnroll.net/latest/installation/setup-ide.html#setup-visual-studio-2022>
- [6] “Guía completa para pruebas basadas en el comportamiento con Pytest BDD | Pytest con Eric.” Accessed: Aug. 29, 2025. [Online]. Available: <https://pytest-with-eric.com/bdd/pytest-bdd/>
- [7] “PyTest — BDD (Behavioural Driven Development) | by Ramkumar R | Medium.” Accessed: Aug. 29, 2025. [Online]. Available: <https://medium.com/@ramanish1992/pytest-bdd-behavioural-driven-development-a5df4d90619a>