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 especifica, y por último “Then (Entonces)” un resultado determinado u una secuencia prevista [3].

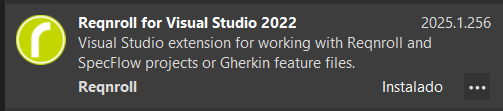
**INVESTIGAR EL ACOPLAMIENTO DE CUCUMBER (LENGUAJE GHERKIN) CON LA PLATAFORMA DE C#**

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](https://go.reqnroll.net/vs2022-extension) [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*



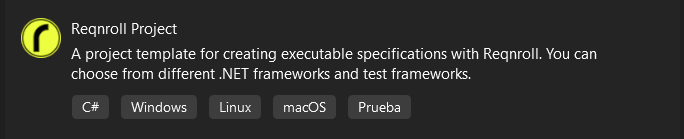
***Tabla 1:*** *Extensión* [Reqnroll para Visual Studio 2022](https://go.reqnroll.net/vs2022-extension)

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

* **ESCRITURA DE ESCENARIOS**

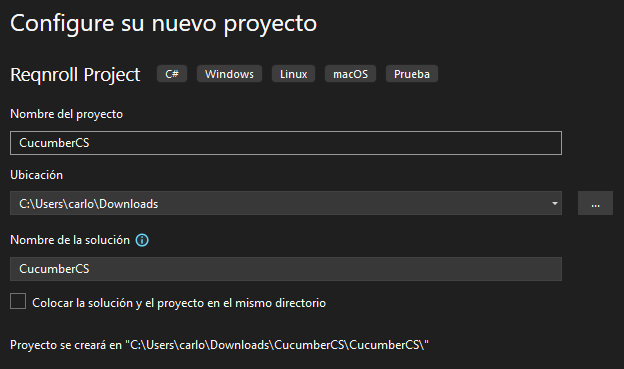
A continuación, se redactará como escribir escenarios en C#. Estos pasos son guiados por videos del canal de YouTube [*Execute Automation*](https://www.youtube.com/@ExecuteAutomation)*.*

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



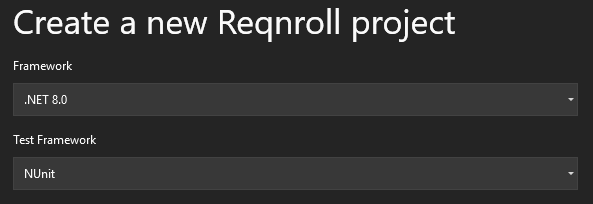
***Imagen 2:*** *Plantilla de proyecto Reqnroll Project*

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



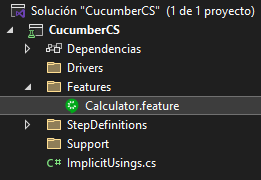
***Imagen 3:*** *Dar un nombre al proyecto*

1. 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.



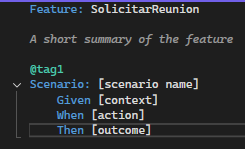
***Imagen 5:*** *Estructura de carpetas creada automáticamente*

1. 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*

1. 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>

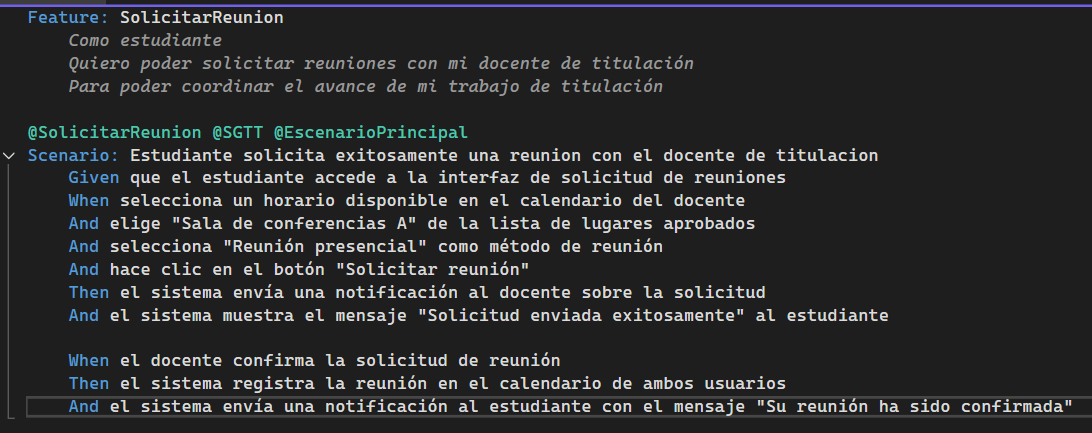
@tag  
Scenario: <título del caso>  
Given <contexto inicial>

When<Acción o evento>

Then<Resultados>

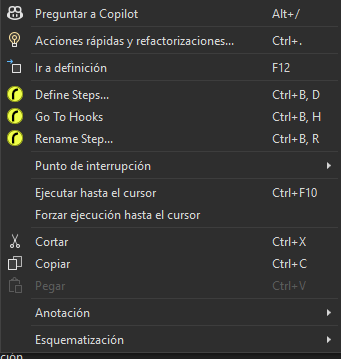
Opcional: And / But para concatenar pasos sin repetir palabras clave.

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



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

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



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

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

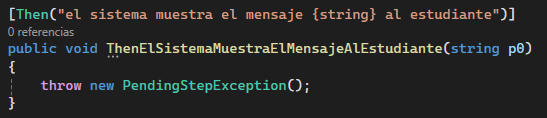


***Imagen 10:*** *Crear definición*

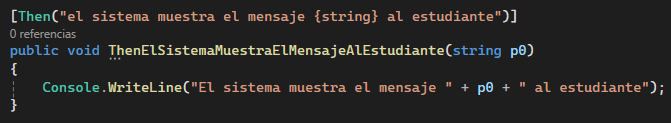
Esto creará un archivo .cs en la carpeta *StepDefinitions* que contendrá un esqueleto con todos los pasos definidos.

* **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.



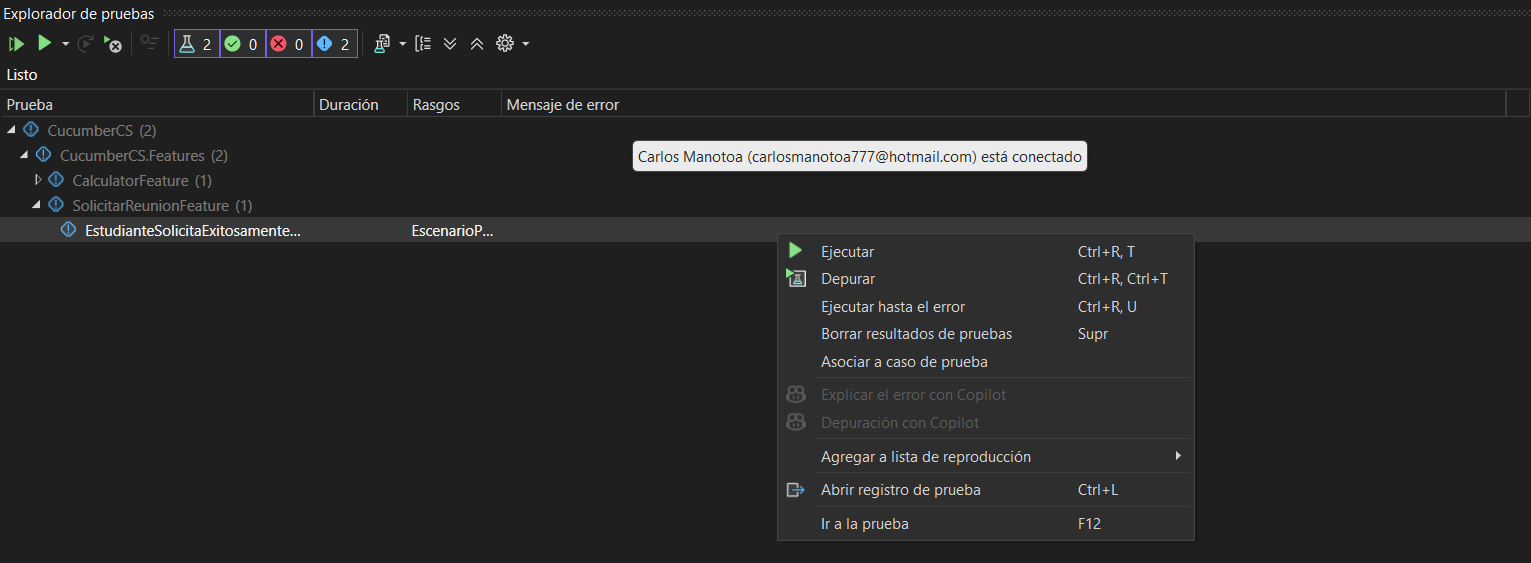
***Imagen 11:*** Código con definición de paso pendiente



***Imagen 12:*** *Definición de paso pendiente reemplazado por Console.WriteLine(“”)*

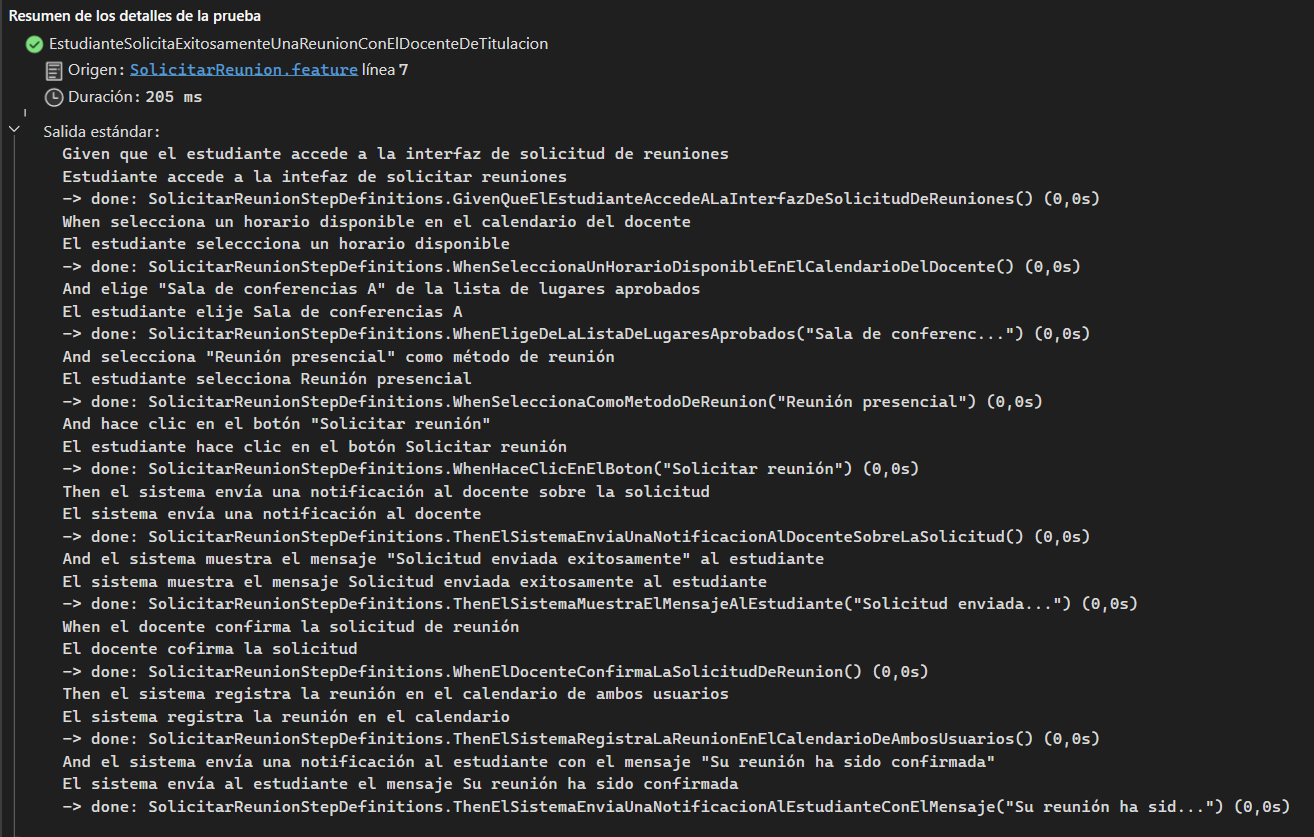
1. Después de reemplazar todas las definiciones de paso pendiente, dar clic sobre *prueba → Explorador de pruebas*

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”

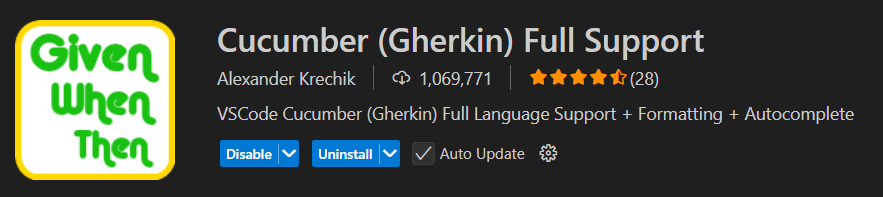
**DEMOSTRAR LA AUTOMATIZACIÓN DE PRUEBAS EN CUCUMBER + PYTHON**

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

* **Instalación**

1. Instalar [Python](https://www.python.org/downloads/)
2. Verificar que Python se instaló correctamente con el comando P*ython –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*

1. Crear la carpeta raíz e ingresar

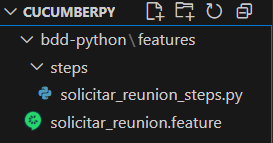
* Crear la carpeta raíz con el comando *mkdir <nombre>*

**Ejemplo:** mkdir bdd-python

* Ingresar dentro de la carpeta raíz usando *cd <nombre>*

**Ejemplo:** cd bdd-python

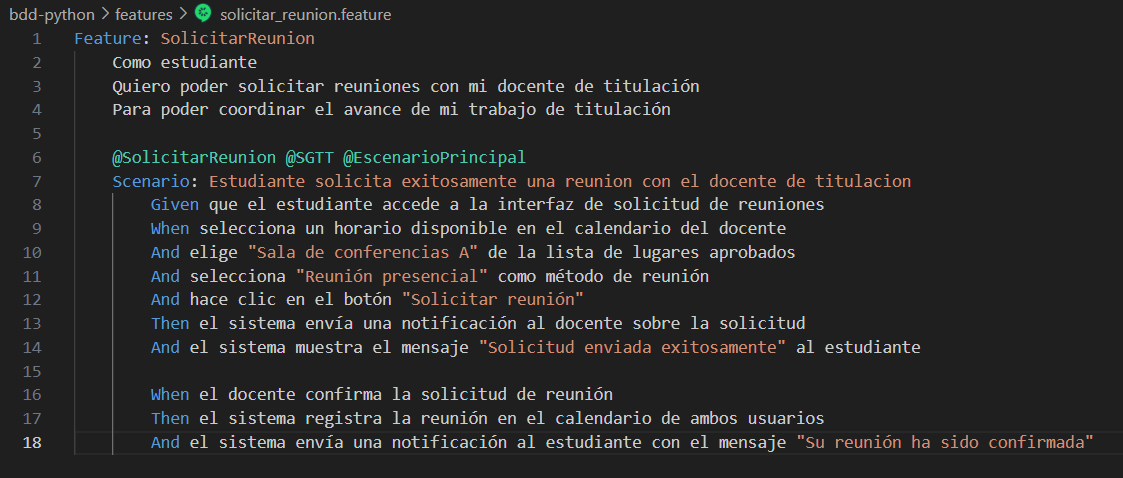
1. Dentro del proyecto crear una carpeta llamada “features” donde se redactarán los escenarios con la extensión .feature
2. Crear una carpeta llamada “Steps” donde se definirá los pasos .py
3. Da Dentro de la carpeta “features” crear un archivo. Features con el nombre del escenario

****

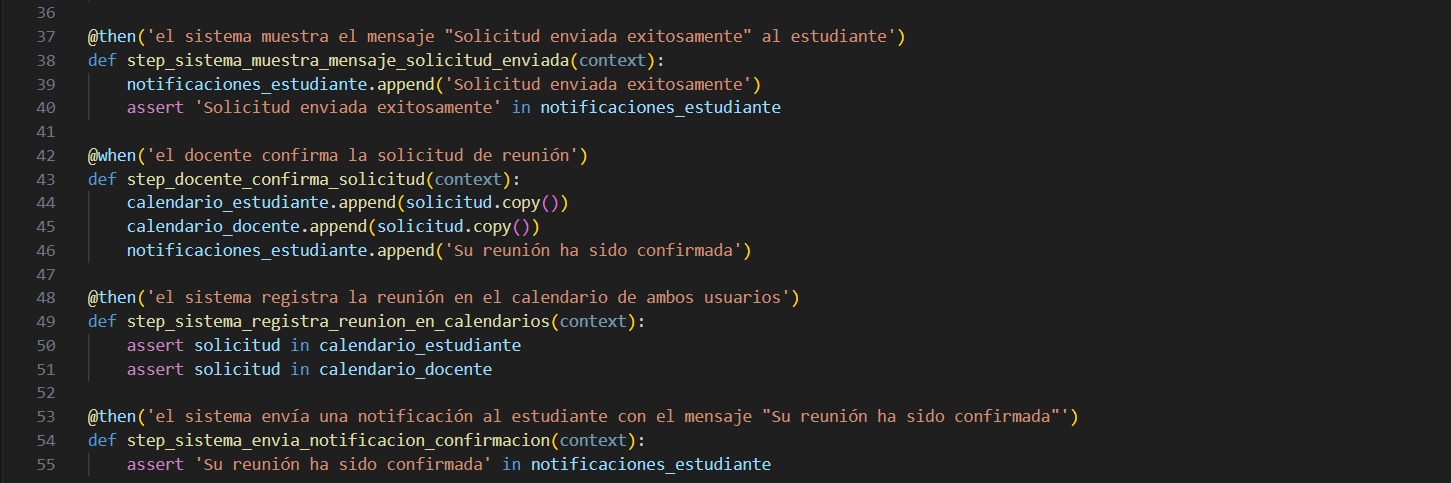
***Imagen 16:*** *Estructura del programa*

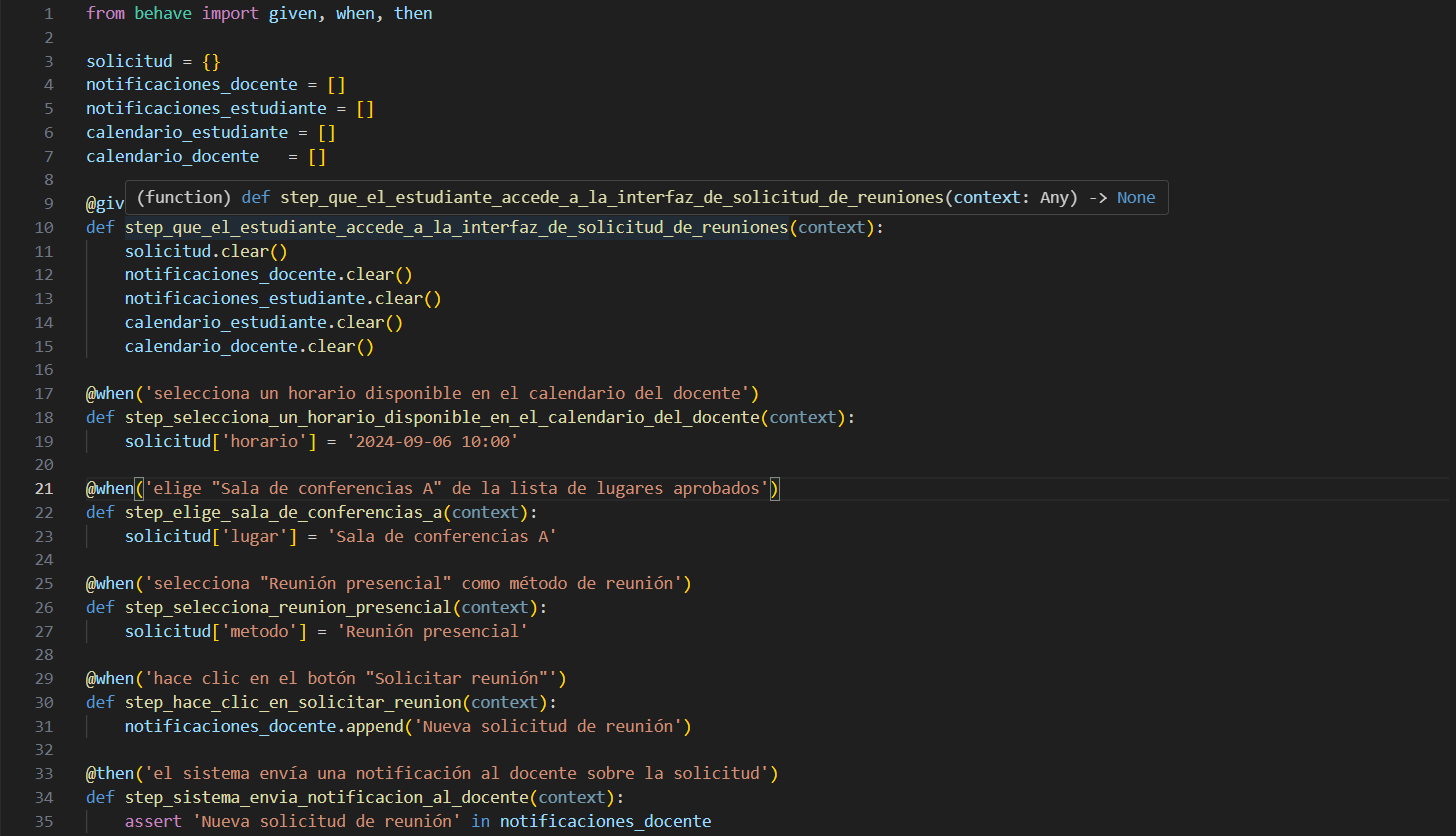
* **Escritura de escenarios**

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



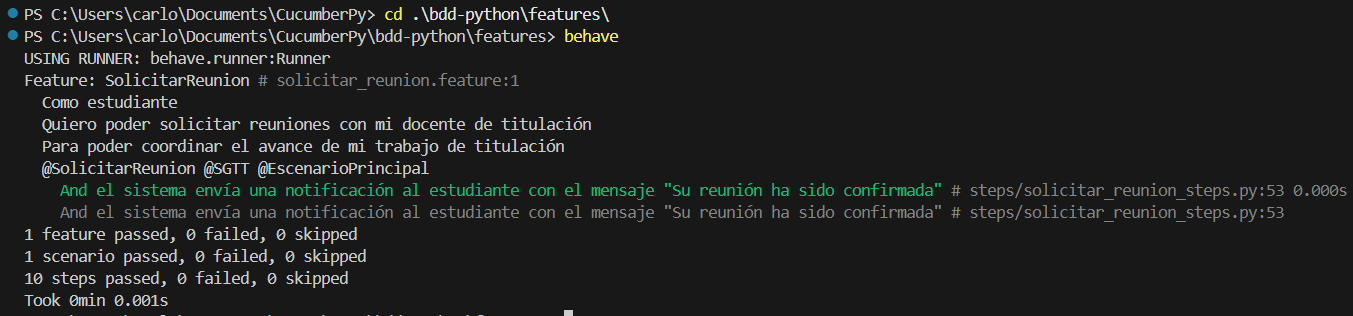
***Imagen 17:*** *Descripción del escenario en Visual Studio Code*

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

***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”



***Imagen 19:*** Ejecución en Visual Studio Code