

ACOPLAMIENTO DE CUCUMBER (LENGUAJE GHERKIN)

Luis Alberto Paca Pilataza

INTRODUCCIÓN

El Desarrollo Dirigido por Comportamiento (BDD) es una metodología ágil que se ha convertido en un pilar para la colaboración en el desarrollo de software. Su propósito principal es crear una conexión clara y compartida entre los requisitos de negocio y las pruebas automatizadas, utilizando un lenguaje común y comprensible para todos los miembros del equipo, desde analistas de negocio hasta desarrolladores[1]. Este lenguaje, conocido como Gherkin, emplea una sintaxis sencilla de Given, When, Then para describir los escenarios de comportamiento de la aplicación[2].

En el ecosistema de C#, la integración de Gherkin se realiza principalmente a través de frameworks como Reqnroll. Aunque SpecFlow fue la herramienta dominante en el pasado, ha llegado a su fin de vida (EoL), y Reqnroll se ha posicionado como su sucesor de código abierto, activamente mantenido por la comunidad y compatible con las últimas versiones de .NET. El acoplamiento técnico se logra mediante "definiciones de pasos" en C#, que son funciones que vinculan las oraciones en Gherkin con la lógica de automatización [3] [4].

Por otro lado, en la plataforma Python, la herramienta BDD más utilizada es Behave, que es un framework diseñado como un "clon" de Cucumber para este lenguaje. Behave también utiliza archivos Gherkin para la descripción de los escenarios y los asocia a código Python mediante "implementaciones de pasos". Este enfoque permite demostrar la automatización de pruebas tanto para interfaces de usuario (UI) web, con bibliotecas como Selenium, como para API REST, usando la librería [5] [6].

Esta portabilidad del lenguaje Gherkin entre plataformas resalta el valor de la metodología BDD, que se centra en el "qué" (el comportamiento del sistema) y permite que la implementación técnica del "cómo" (el código de automatización) se adapte a cualquier tecnología [7].

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

Concepto de BDD y Gherkin

BDD

Metodología que conecta requisitos de negocio con pruebas automatizadas, usando un lenguaje comprensible para todos (analistas, testers, desarrolladores) [7] [8].

Gherkin

Lenguaje de especificación ejecutable. Define escenarios con palabras clave Given, When, Then, y opcionales And y But. Permite separar el "qué" (comportamiento esperado) del "cómo" (implementación en C#)[2] [9].

Herramientas para BDD en C#

SpecFlow

Fue la herramienta principal para BDD en .NET, pero está en fin de vida (EoL) y ya no se considera de código abierto activamente mantenida [10] [11].

Reqnroll

Su sucesor recomendado, activo y mantenido por la comunidad. Es compatible con las últimas versiones de .NET (incluido .NET 8.0) y se integra con frameworks de pruebas como NUnit, MSTest y xUnit [11] [12] .

Acoplamiento técnico entre Gherkin y C#

Step Definitions

Clases C# que vinculan los pasos de Gherkin con la lógica de automatización. Los métodos se asocian con atributos como [4].

[Given], y. Los parámetros se pueden definir usando expresiones de Cucumber ({string}, {int}, {word}) o expresiones regulares.

Manejo de datos y contexto

Se pueden pasar datos a las definiciones de pasos a través de parámetros en línea, tablas de datos (Data Tables) y el objeto ScenarioContext. El ScenarioContext es útil para compartir objetos, como una instancia de IWebDriver, entre diferentes pasos del mismo escenario [13].

Migración de SpecFlow a Reqnroll

Migración rápida

Consiste en usar el paquete de compatibilidad Reqnroll.SpecFlowCompatibility para mantener los espacios de nombres antiguos (TechTalk.SpecFlow) con cambios mínimos en el código [14].

Migración completa

Requiere reemplazar todos los paquetes NuGet de SpecFlow.* por sus equivalentes Reqnroll.*, cambiar los espacios de nombres a Reqnroll y renombrar el archivo de configuración specflow.json a reqnroll.json [14].

INSTALACIÓN

Paso 1: Abrir Visual Studio 2022

- Asegúrate de tener una versión compatible de Visual Studio 2022 (Community, Professional o Enterprise).

Paso 2: Abrir el Administrador de Extensiones

- En la barra de menús superior, haz clic en "Extensiones".
- Selecciona "Administrar extensiones".

Paso 3: Buscar la Extensión ReqnRoll

- En la ventana del Administrador de extensiones:
 - Selecciona la pestaña "Examinar" (Browse).
 - En el cuadro de búsqueda (esquina superior derecha), escribe "ReqnRoll".

- Debería aparecer "ReqnRoll for Visual Studio 2022" (como se muestra en tu ejemplo).

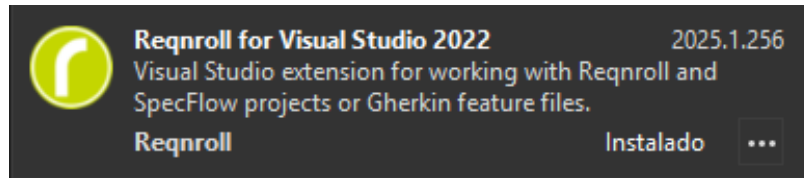


Figura 1: Instalación de la extensión ReqnRoll en Visual Studio 2022

Paso 4: Descargar e Instalar

- Haz clic en "ReqnRoll for Visual Studio 2022".
- Presiona el botón "Descargar" (Download).
- La extensión se descargará y programará para instalarse.

Paso 5: Reiniciar Visual Studio

- Cierra completamente Visual Studio 2022 para que la instalación se complete.
- Al reiniciar, la extensión estará activa.

ESCRITURA DE ESCENARIOS

Paso 1: Crear un nuevo proyecto de ReqnRoll en Visual Studio 2022

1. Abre Visual Studio 2022.
2. Haz clic en "Crear un nuevo proyecto".
3. Busca la plantilla "Reqnroll Project" (debe aparecer después de instalar la extensión).
4. Selecciona la plantilla y haz clic en "Siguiente".

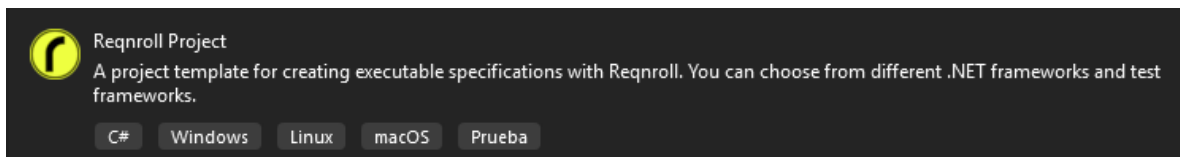
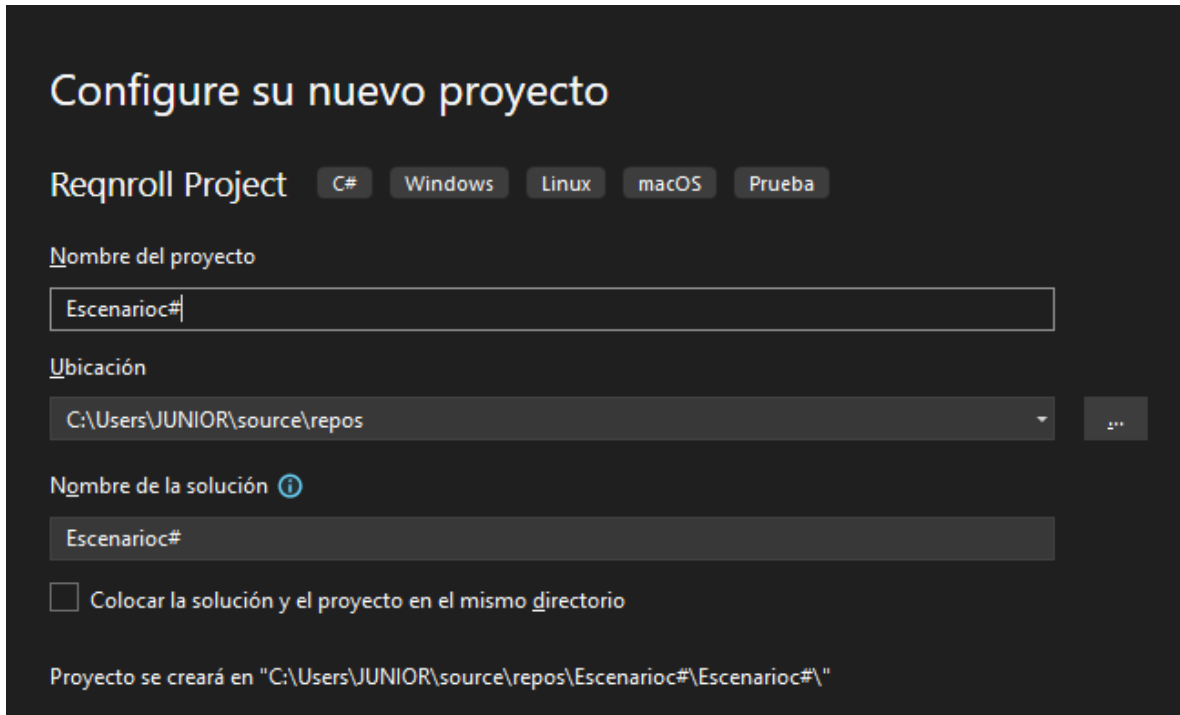


Figura 2: Selección de la plantilla ReqnRoll Project

Paso 2: Configurar el proyecto

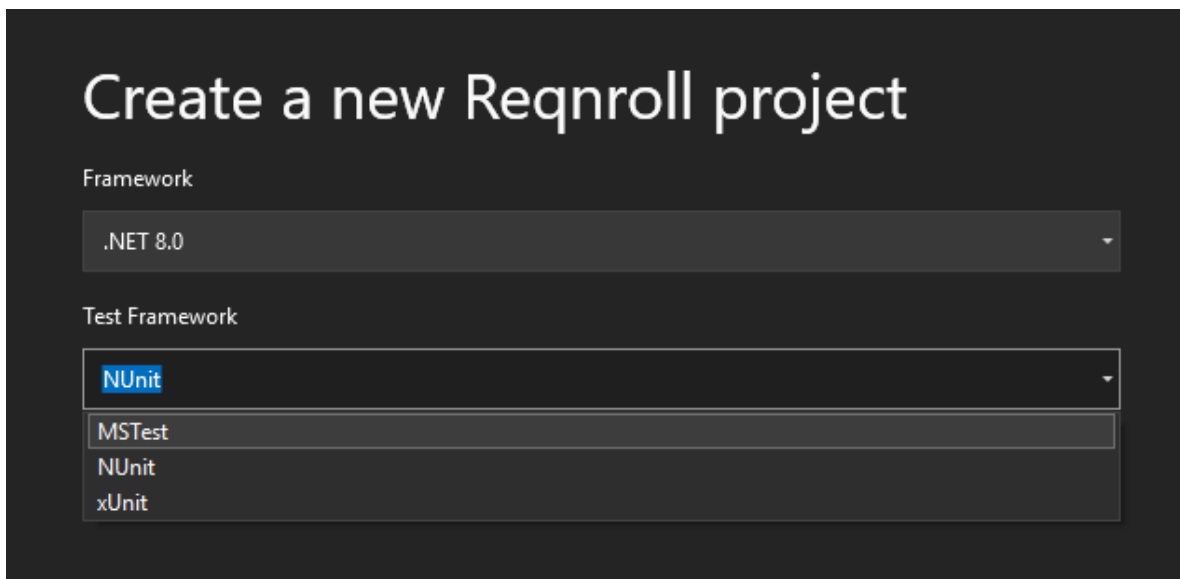
1. Asigna un nombre al proyecto (ej: Reqnroll.Tests).



The screenshot shows the 'Configure your new project' window. At the top, it says 'Reqnroll Project' with tabs for 'C#', 'Windows', 'Linux', 'macOS', and 'Prueba'. Below this, there are three main sections: 'Nombre del proyecto' with a text input field containing 'Escenarioc#'; 'Ubicación' with a dropdown menu showing 'C:\Users\JUNIOR\source\repos' and a folder icon; and 'Nombre de la solución' with a text input field containing 'Escenarioc#'. There is an unchecked checkbox labeled 'Colocar la solución y el proyecto en el mismo directorio'. At the bottom, it states 'Proyecto se creará en "C:\Users\JUNIOR\source\repos\Escenarioc#\Escenarioc#\''.

Figura 3: Configuración del nombre del proyecto

2. Selecciona:
 - Framework: .NET 8.0 o superior.
 - Test Framework: NUnit (o xUnit si prefieres).



The screenshot shows the 'Create a new Reqnroll project' window. It has two main sections: 'Framework' with a dropdown menu showing '.NET 8.0'; and 'Test Framework' with a dropdown menu showing 'NUnit' selected, and a list of other options: 'MSTest', 'NUnit', and 'xUnit'.

Figura 4: Selección del framework y test framework

3. Haz clic en "Crear".

Paso 3: En explorador de soluciones

La plantilla crea automáticamente:

- Una carpeta Features (contiene archivos.feature).
- Una carpeta StepDefinitions (contiene clases para definir los pasos).
- Un archivo de ejemplo Calculadora.feature (puede eliminarlo o usarlo como referencia).

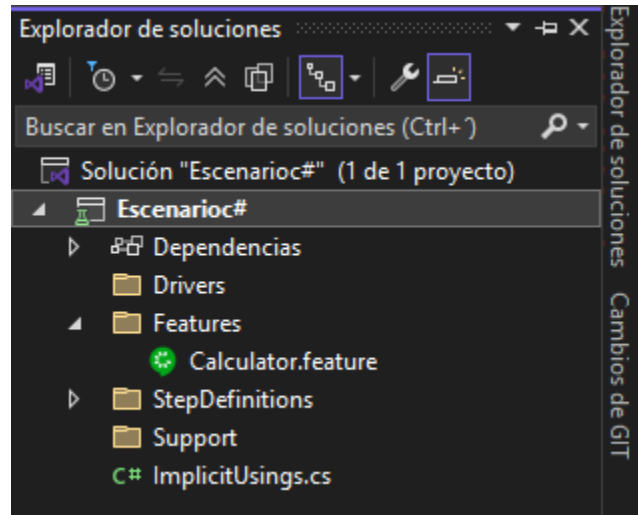


Figura 5: Estructura de carpetas generada automáticamente

Paso 4: Agregar un nuevo archivo.feature

1. Haz clic derecho en la carpeta Features.
2. Selecciona Agregar → Nuevo elemento.
3. Busca la plantilla "Feature File for Reqnroll".
4. Asigna un nombre (ej: Registro de Anteproyecto.feature).



Figura 7: Plantilla automática de un archivo.feature

5. Haz clic en "Agregar".

Paso 5: Escribir el escenario en Gherkin

1. El archivo.feature tendrá una plantilla básica.

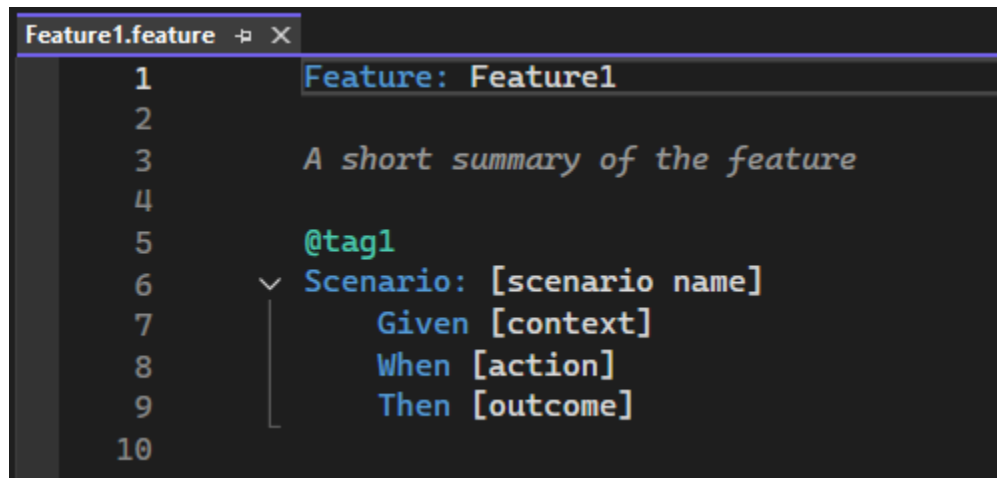


Figura 9: Generación de step definitions

2. Editar con la estructura Gherkin.

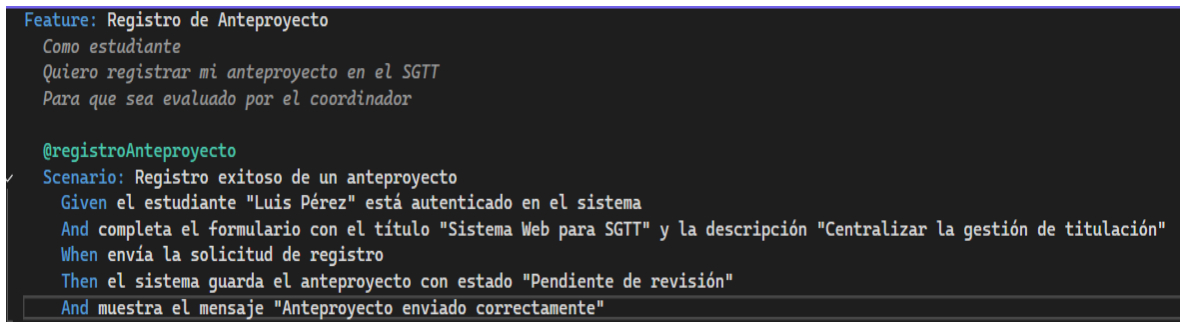


Figura 10: Ventana de creación de step definitions

Paso 6: Compilar y generar las definiciones de pasos (Step Definitions)

1. Compila el proyecto (Ctrl + Shift + B).
2. En el archivo.feature, haz clic derecho sobre una palabra clave (ej: Given).
3. Selecciona "Define Steps" (o "Generar Step Definitions").

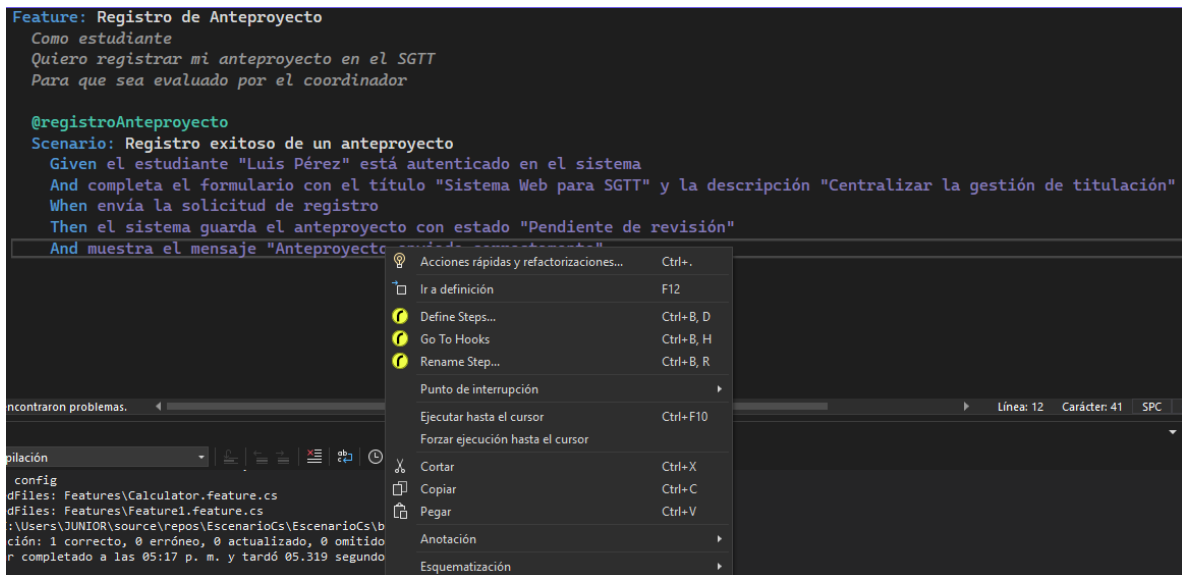
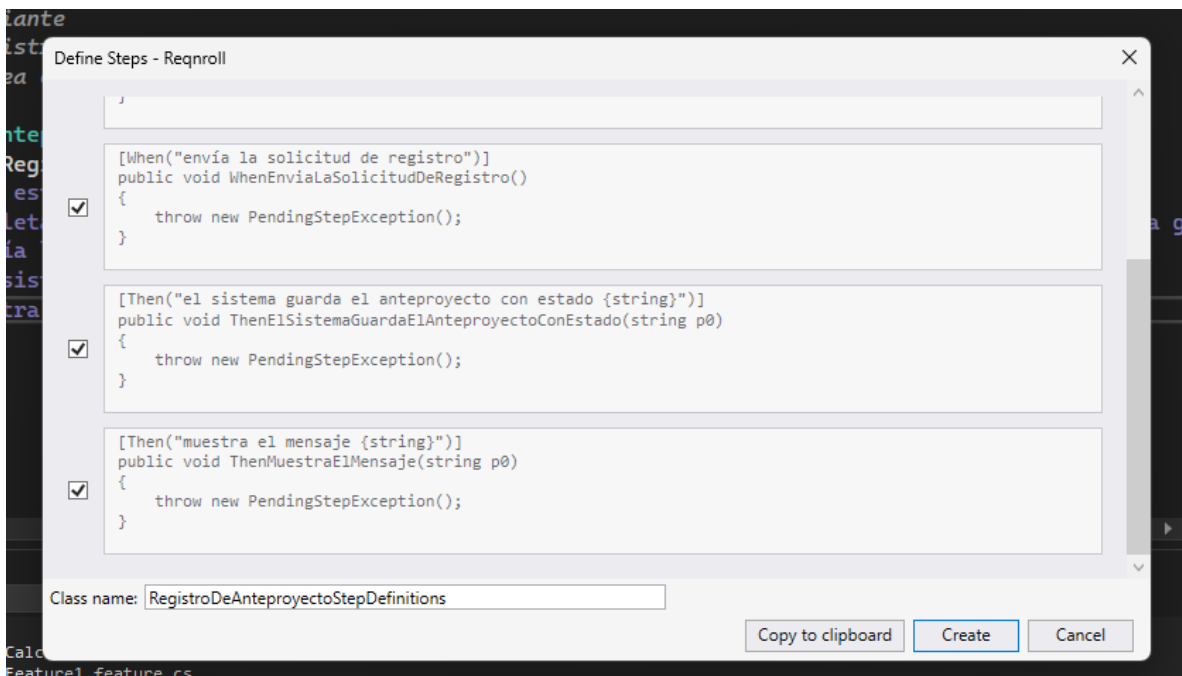


Figura 11: Código con step definitions pendientes

4. En la ventana emergente, selecciona todos los pasos y haz clic en "Create".



Paso 7: Implementar los step definitions

- ReqnRoll creará una clase en la carpeta StepDefinitions con métodos vacíos en C#.

PRUEBA Y VALIDACIÓN

Paso 1: Reemplazar definiciones de paso pendiente por Console.WriteLine

Cuando generas los step definitions automáticamente, ReqRoll crea métodos con una excepción de "pendiente" (PendingStepException). Para evitar errores y validar rápidamente el flujo, puedes reemplazar temporalmente estas excepciones con Console.WriteLine.

Ejemplo antes (código pendiente):

```
[Then("muestra el mensaje {string}")]
0 referencias
public void ThenMuestraElMensaje(string p0)
{
    throw new NotImplementedException();
}
```

Figura 12: Step definitions con Console.WriteLine

Ejemplo después (con Console.WriteLine):

```
[Then("muestra el mensaje {string}")]
0 referencias
public void ThenMuestraElMensaje(string p0)
{
    Console.WriteLine("muestra el mensaje " + p0);
}
```

Figura 13: Step definitions con Console.WriteLine

Paso 2: Abrir el Explorador de Pruebas

1. En la barra de menús de Visual Studio, ve a "**Prueba**" (Test).
2. Selecciona "**Explorador de pruebas**" (Test Explorer).
3. También puedes usar el atajo: **Ctrl+E, T**.

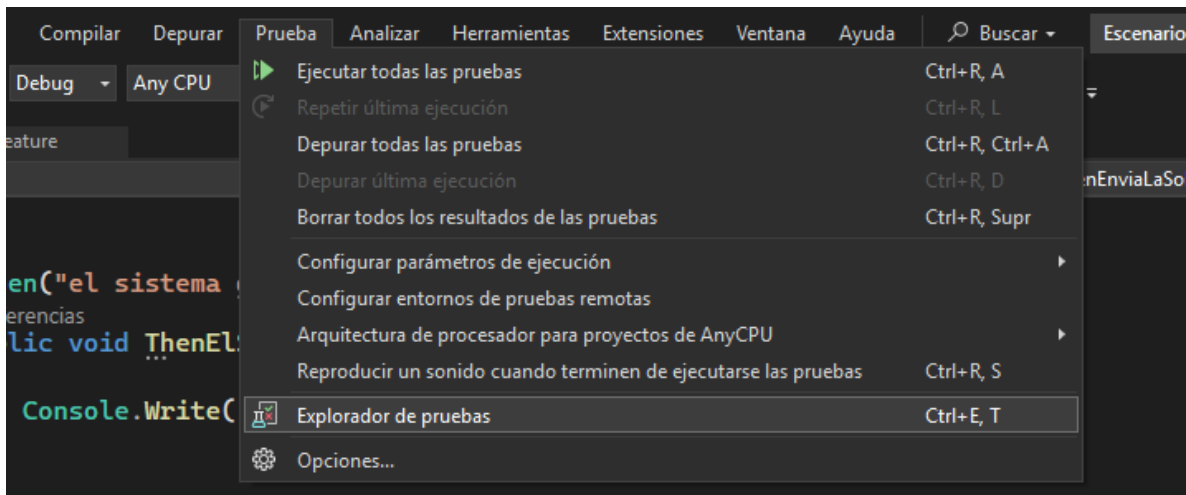


Figura 14: Explorador de pruebas de Visual Studio

Paso 3: Ejecutar los escenarios

1. En el Explorador de pruebas, verás todos los escenarios de Gherkin listados (agrupados por features).
2. Haz clic derecho en el escenario que quieres ejecutar
3. Selecciona **"Ejecutar"** (Run).

Paso 4: Ver los resultados de la ejecución

- Si todos los pasos tienen Console.WriteLine, verás en la salida de consola cada paso impreso.
- El resultado en el Explorador de pruebas mostrará "Aprobado" (green check) porque ya no hay excepciones pendientes.
- Esto confirma que el flujo del escenario se está ejecutando en el orden correcto.

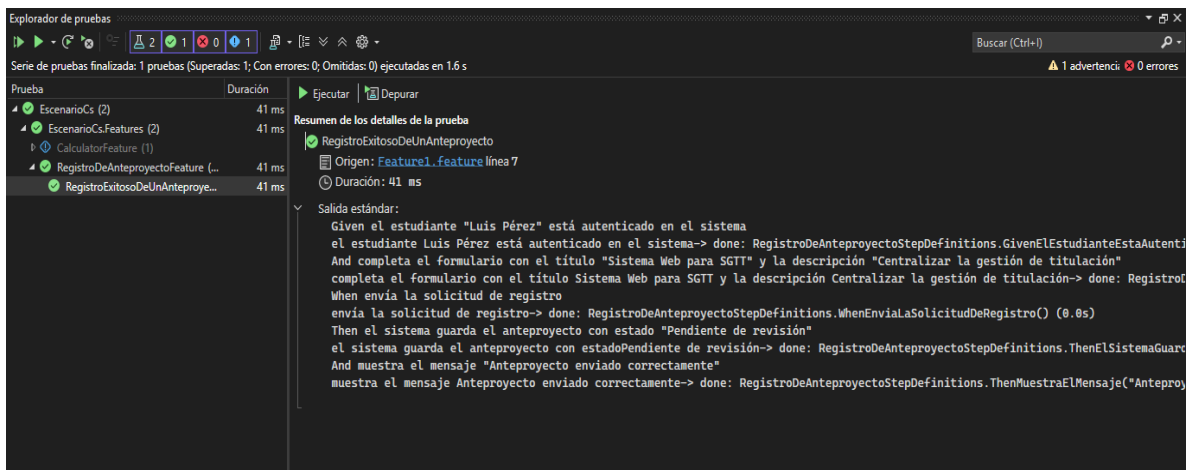


Figura 15: Resultado de la ejecución de pruebas

AUTOMATIZACIÓN DE PRUEBAS EN CUCUMBER + PYTHON

Concepto de BDD y Gherkin

BDD (Behavior Driven Development)

El BDD es una metodología de desarrollo de software ágil que se enfoca en la colaboración y la comunicación entre los equipos técnicos y no técnicos, como desarrolladores, testers y analistas de negocio. Utiliza un lenguaje común, como Gherkin, para definir el comportamiento esperado de un sistema desde la perspectiva del usuario. Esto crea una "documentación viva" del sistema que es comprensible para todos [5].

Gherkin

Gherkin es el lenguaje de texto plano que utilizan los frameworks BDD como Behave para estructurar las especificaciones ejecutables. Su sintaxis se centra en un patrón de tres actos:

Given (Dado): Establece el contexto o el estado inicial del sistema.

When (Cuando): Describe la acción o el evento que realiza el usuario.

Then (Entonces): Define el resultado o la aserción esperada después de la acción.

Las palabras clave And y But se utilizan para añadir pasos adicionales a las cláusulas Given, When o Then para mejorar la legibilidad del escenario. El uso de Background permite definir pasos comunes que se ejecutan antes de cada escenario en un mismo archivo, reduciendo la redundancia [2][15].

Herramientas para BDD en Python

Behave es la implementación de referencia y el framework de código abierto más popular para el BDD en Python. Utiliza la misma sintaxis Gherkin que otros frameworks de la familia Cucumber y se integra con bibliotecas populares para la automatización, como Selenium para pruebas de interfaz de usuario (UI) web y requests para pruebas de API. Su configuración es sencilla, ya que busca automáticamente las pruebas en una estructura de directorios predefinida, y tiene un ejecutor de pruebas integrado, a diferencia de otros frameworks que requieren plugin adicionales [13][6].

Alternativas a Behave

El ecosistema de Python ofrece otras herramientas de BDD, como Pytest BDD y Lettuce, que también son opciones comunes [1]. Pytest BDD es una extensión del framework de pruebas Pytest, lo que le permite aprovechar su ecosistema de más de 800 plugins y su capacidad de ejecución paralela de forma nativa. Por otro lado, Lettuce es similar a Behave, aunque se considera más adecuado para proyectos BDD de menor escala debido a su menor adopción y a su limitada comunidad [16].

Acoplamiento técnico entre Gherkin y Python

Step Definitions

La conexión entre los escenarios de Gherkin y el código de Python se realiza a través de las "definiciones de pasos", que son funciones de Python con decoradores especiales (@given, @when, @then). Estas funciones deben estar en un archivo [16]. Dentro de la carpeta features/steps/. La lógica de la prueba se implementa dentro de estas funciones, vinculando el texto del paso de Gherkin con el código ejecutable de Python [6]. El objeto context de Behave es crucial, ya que se pasa como

argumento a cada función y sirve como un almacén de datos para compartir información entre diferentes pasos dentro del mismo escenario [5].

Behave ofrece ganchos (hooks) que permiten ejecutar código de configuración y limpieza en momentos específicos del ciclo de vida de la prueba. Estos ganchos se definen en un archivo opcional llamado `environment.py`. Los más comunes son `before_all` y `after_all` (que se ejecutan una vez por ejecución completa), y `before_scenario` y `after_scenario` (que se ejecutan antes y después de cada escenario) [17]. Estos ganchos son esenciales para inicializar recursos como el WebDriver de Selenium y asegurar que se cierren correctamente para evitar fugas de memoria, lo que mantiene el marco de pruebas estable y fiable [18].

INSTALACIÓN

Paso 1: Instalar Python

- Descarga Python desde python.org.
- Ejecuta el instalador y marca la opción "Add Python to PATH".
- Verifica la instalación:

```
bash
```

```
python --version
```

Debe mostrar la versión instalada (ej: Python 3.12.0).

Paso 2: Instalar Behave

- Abre una terminal y ejecuta:

```
bash
```

```
pip install behave
```

- Verifica la instalación:

```
bash
```

```
behave --version
```

Debe mostrar la versión (ej: behave 1.2.6).

Paso 3: Instalar la extensión en VS Code

- Abre VS Code y ve a Extensiones (Ctrl+Shift+X).
- Busca: "Cucumber (Gherkin) Full Support".
- Busca: "C# Dev Kit por compatibilidad"
- Instálala y reinicia VS Code.

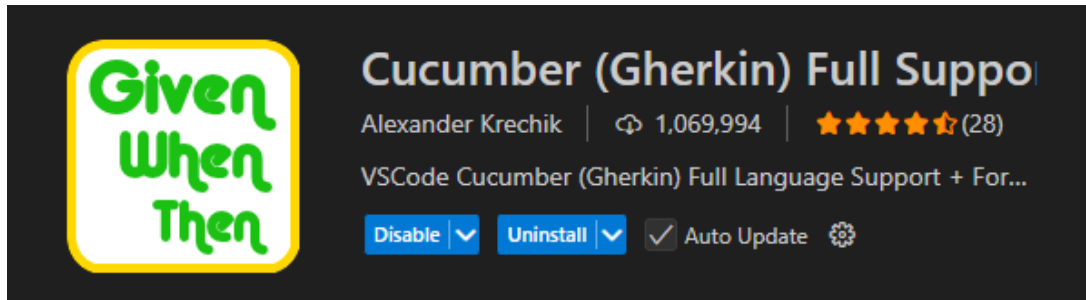


Figura 16: Extensión Cucumber (Gherkin) Full Support en VS Code

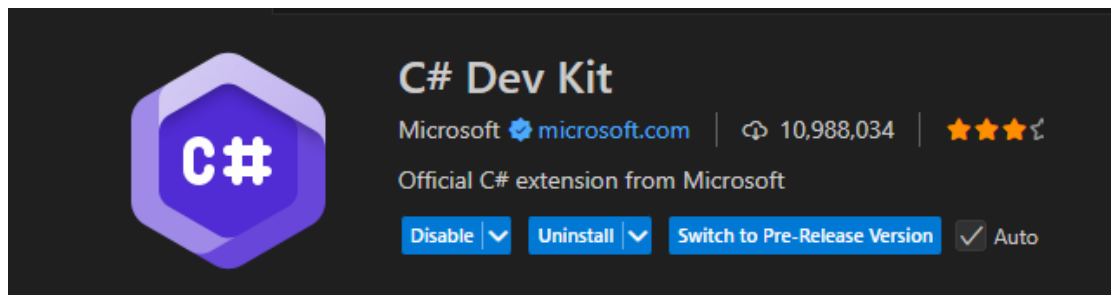


Figura 17: Extensión C# Dev Kit en VS Code

Paso 4: Crear la estructura de carpetas

- En la terminal, ejecuta:

```
mkdir bdd-python
```

```
cd bdd-python
```

```
mkdir features
```

```
mkdir features/steps
```

ESCRITURA DE ESCENARIOS

Paso 5: Crear archivo .feature

- En la carpeta features, crea un archivo llamado...
- Escribe el escenario en Gherkin:

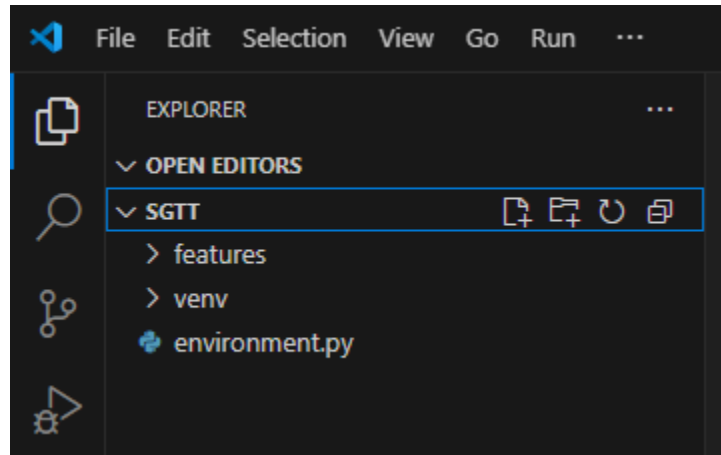


Figura 18: Estructura de carpetas en Python

Paso 6: Crear step definitions

En la carpeta features/steps, crea un archivo steps.py.

Implementa los steps:

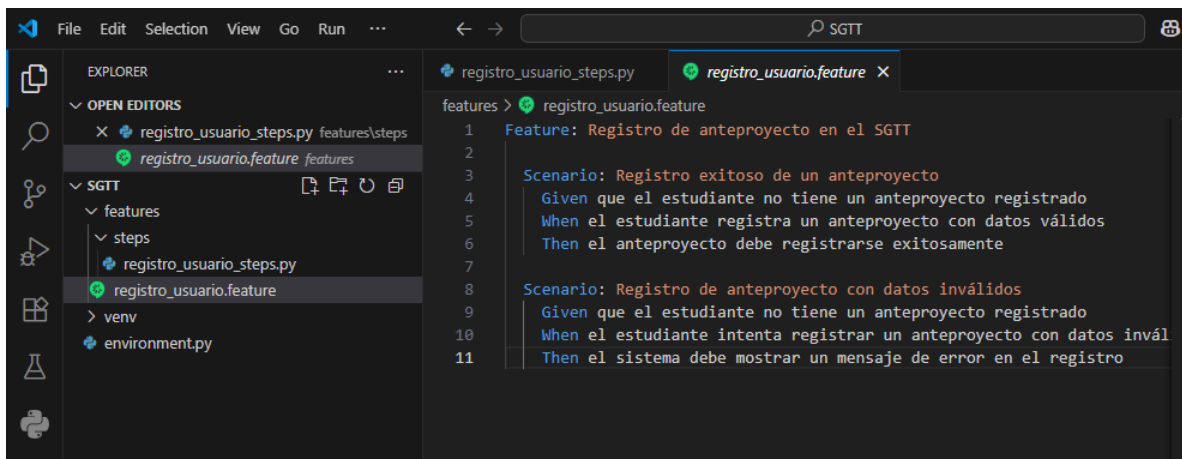


Figura 19: Escenario Gherkin en VS Code

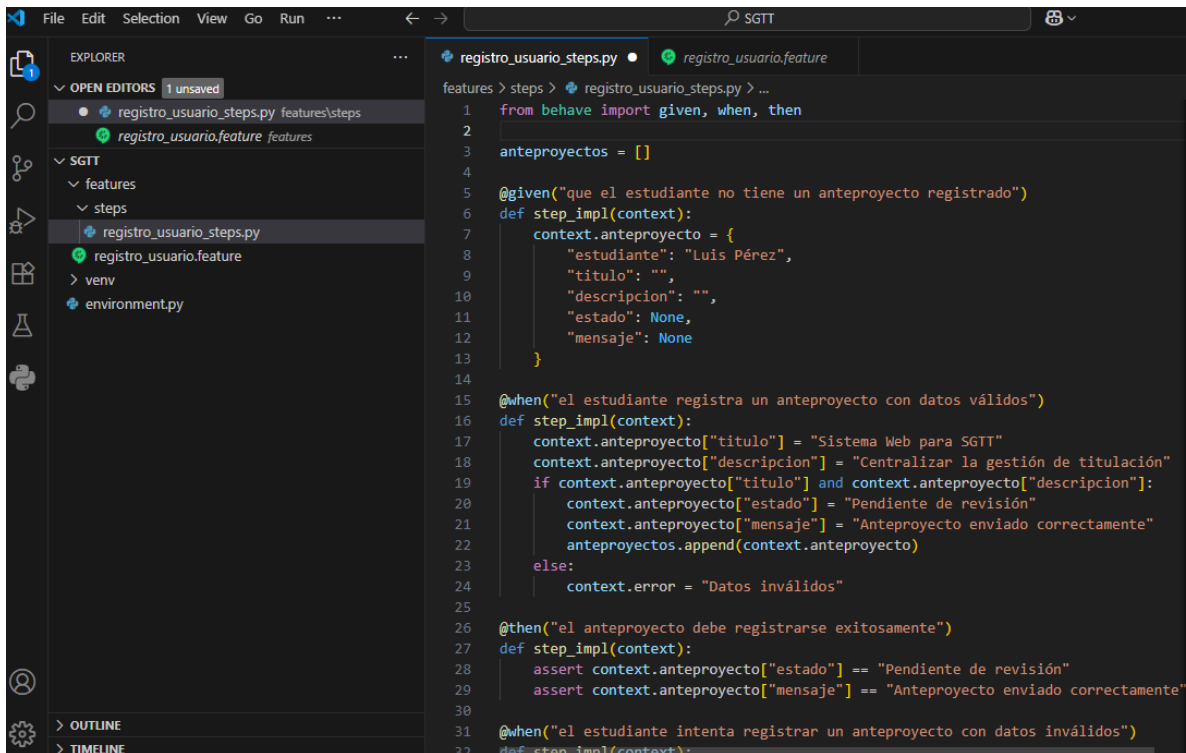


Figura 20: Step definitions en Python

PRUEBA Y VALIDACIÓN

Paso 7: Ejecutar las pruebas

- Abre la terminal en VS Code (Ctrl+Shift+ù).
- Navega a la carpeta raíz bdd-python.
- Ejecuta: behave en CLI

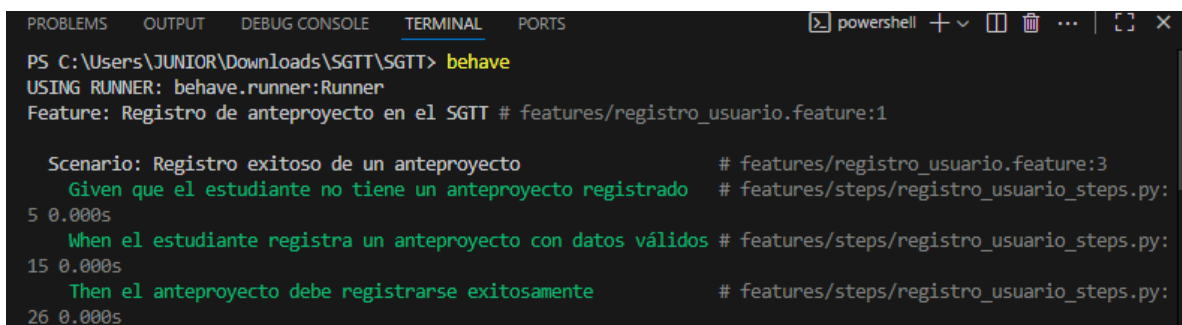


Figura 21: Ejecución de pruebas con Behave en terminal

BIBLIOGRAFIA

- [1] "Pytest BDD vs Behave: Pick the Best Python BDD Framework - Codoid." Accessed: Aug. 30, 2025. [Online]. Available: <https://codoid.com/automation-testing/pytest-bdd-vs-behave-pick-the-best-python-bdd-framework/>
- [2] "Reference | Cucumber." Accessed: Aug. 30, 2025. [Online]. Available: <https://cucumber.io/docs/gherkin/reference/>
- [3] "SpecFlow ya no es open source : r/dotnet." Accessed: Aug. 30, 2025. [Online]. Available: https://www.reddit.com/r/dotnet/comments/1hgyxrp/specflow_is_no_longer_open_source/
- [4] "Step Definitions - Reqnroll Documentation." Accessed: Aug. 30, 2025. [Online]. Available: <https://docs.reqnroll.net/latest/automation/step-definitions.html>
- [5] "Advanced Guide to Behavior-Driven Development with Behave in Python | by Moraneus | Medium." Accessed: Aug. 30, 2025. [Online]. Available: <https://medium.com/@moraneus/advanced-guide-to-behavior-driven-development-with-behave-in-python-aaa3fa5e4c54>
- [6] "Rest API testing using behave in Python | Globalnow IT Services." Accessed: Aug. 30, 2025. [Online]. Available: <https://www.globalnowit.com/rest-api-testing-using-behave-in-python/>
- [7] "About Reqnroll | Reqnroll." Accessed: Aug. 30, 2025. [Online]. Available: <https://reqnroll.net/about/>
- [8] "Cucumber." Accessed: Aug. 30, 2025. [Online]. Available: <https://cucumber.io/>
- [9] "Writing scenarios with Gherkin syntax | CucumberStudio Documentation." Accessed: Aug. 30, 2025. [Online]. Available: <https://support.smartbear.com/cucumberstudio/docs/bdd/write-gherkin-scenarios.html>
- [10] "Specflow C# (Cómo funciona para desarrolladores)." Accessed: Aug. 30, 2025. [Online]. Available: <https://ironpdf.com/es/blog/net-help/specflow-csharp/>
- [11] "SpecFlow ya no es open source : r/dotnet." Accessed: Aug. 30, 2025. [Online]. Available: https://www.reddit.com/r/dotnet/comments/1hgyxrp/specflow_is_no_longer_open_source/
- [12] "SpecFlow to Reqnroll: A Step-by-Step Migration Guide - Codoid." Accessed: Aug. 30, 2025. [Online]. Available: <https://codoid.com/automation-testing/specflow-to-reqnroll-a-step-by-step-migration-guide/>
- [13] "Specflow: Test Automation — BDD and C# — Step by step guide (2024) | by Naveen Kolambage | Medium." Accessed: Aug. 30, 2025. [Online]. Available: <https://medium.com/@naveenkolambage/specflow-test-automation-bdd-and-c-step-by-step-guide-2024-d4b7badfac73>
- [14] "Migrating from SpecFlow - Reqnroll Documentation." Accessed: Aug. 30, 2025. [Online]. Available: <https://docs.reqnroll.net/latest/guides/migrating-from-specflow.html>

- [15] "Behave - Scenario Outlines." Accessed: Aug. 30, 2025. [Online]. Available: https://www.tutorialspoint.com/behave/behave_scenario_outlines.htm
- [16] "Top Python Testing Frameworks in 2025 - TestGrid." Accessed: Aug. 30, 2025. [Online]. Available: <https://testgrid.io/blog/python-testing-framework/>
- [17] "Behave - Hooks." Accessed: Aug. 30, 2025. [Online]. Available: https://www.tutorialspoint.com/behave/behave_hooks.htm
- [18] "Advanced usage; extending behave-webdriver — behave-webdriver 0.0.1a documentation." Accessed: Aug. 30, 2025. [Online]. Available: <https://behave-webdriver.readthedocs.io/en/latest/examples.html>