

Evaluación del módulo 5

Consigna del proyecto 

Evaluación del módulo

Proyecto: Escenarios de Comportamiento (BDD)

Situación inicial

Unidad solicitante: Equipo de Aseguramiento de Calidad (QA) de una empresa de desarrollo de software.

El equipo de QA ha recibido la solicitud de implementar un conjunto de pruebas automatizadas basadas en la metodología de Desarrollo Conducido por Comportamiento (BDD). Concretamente, se desea verificar la calidad y el comportamiento de una aplicación web encargada de gestionar distintas funcionalidades (como registro, login, operaciones de CRUD, etc.), asegurando que cada flujo cumpla con los requerimientos definidos por el área de negocio.

Para ello, se ha identificado que muchos de los problemas en producción provienen de una deficiente comunicación entre los equipos de Desarrollo y Negocio, así como de la falta de claridad en la documentación de los requisitos. Ante este panorama, se quiere introducir Cucumber y la escritura de escenarios en Gherkin para lograr una mejor trazabilidad y entendimiento de los requerimientos.

Nuestro objetivo

El objetivo de este proyecto es diseñar y automatizar **escenarios de prueba BDD** que cubran los principales flujos de la aplicación web, utilizando la sintaxis Gherkin y aprovechando las características de Cucumber (hooks, tags, escenarios outline, etc.). Al finalizar, se espera contar con:

1. **Conjunto de archivos `.feature`** con escenarios redactados en Gherkin para reflejar el comportamiento esperado de la aplicación.
2. **Step Definitions** en un lenguaje de programación (ej.: Java, JavaScript o Python) que implementen la lógica de cada escenario.
3. **Uso de Tags y Hooks** para organizar y optimizar la ejecución de las pruebas.

4. **Reporte de ejecución** que demuestre la trazabilidad de dichos escenarios con las historias de usuario o requerimientos, y que evidencie los resultados de cada prueba.

De esta manera, se busca **mejorar la colaboración** entre las áreas de Desarrollo, QA y Negocio, asegurando que todos comprendan y validen los requisitos desde el inicio.

Requerimientos

Para cumplir con el objetivo propuesto, deben considerarse los siguientes requerimientos:

Generales

1. Estructura de archivos:

- Ubicar los archivos `.feature` dentro de una carpeta especial (por ejemplo, `src/test/resources/features` o equivalente) para mantener ordenados los escenarios.
- Ubicar las **Step Definitions** en una carpeta (por ejemplo, `src/test/java/steps`) que centralice la lógica de los pasos.

2. Lenguaje Gherkin:

- Redactar cada escenario siguiendo la sintaxis de Gherkin (Feature, Scenario, Given, When, Then, And, But).
- Aprovechar **Scenario Outline** y la sección **Examples** cuando necesiten repetirse pasos con distintos datos de prueba.

3. Organización con Etiquetas (Tags):

- Utilizar tags como `@SmokeTest`, `@Regression`, `@WIP`, etc., para filtrar los escenarios.
- Aplicar etiquetas específicas (p.ej., `@Login`, `@CRUD`) que ayuden a agrupar escenarios por funcionalidad.

4. Utilización de Hooks:

- Emplear hooks (por ejemplo, `@Before`, `@After`) para inicializar y cerrar el navegador u otro entorno de prueba.

- Centralizar la configuración y limpieza de datos, evitando la repetición de lógica en los steps.

5. Trazabilidad con Historias de Usuario:

- Para cada escenario, debe hacerse referencia a la historia de usuario o al requerimiento funcional que se esté validando.
- Incluir esta referencia en el reporte final (un documento o un archivo de salida) donde se muestre el resultado de la ejecución.

6. Ambiente de Ejecución:

- Se permitirá el uso de **herramientas gratuitas** para el desarrollo, como Visual Studio Code, IntelliJ IDEA Community, Eclipse o similar.
- En caso de necesitar base de datos, se recomienda usar **SQL Online** o algún servicio gratuito que posibilite ejecutar consultas de manera rápida.

Técnicos

1. Compatibilidad con Selenium (opcional):

- En caso de pruebas de interfaz web, integrar Selenium WebDriver (u otra librería de automatización de la UI) dentro de las Step Definitions para interactuar con la aplicación.

2. Configuración de un Runner:

- Crear una clase (por ejemplo, **TestRunner**) que configure la ejecución de los .feature: la ruta, el glue (pasos), el plugin para reportes (pretty, html, json), etc.

Paso a paso

Este proyecto refiere exclusivamente al **módulo 5: Desarrollo** conducido por comportamiento, y se compone de **6 etapas (lecciones)**, las cuales podrás avanzar de forma progresiva y escalonada con la ayuda de los manuales teóricos y los contenidos desarrollados en las clases en vivo.

Considera invertir **tiempo asincrónico** para el desarrollo de cada etapa a modo de poder finalizar el módulo y realizar la entrega formal de tu propuesta.

Cualquier consulta que surja compártela en los espacios sincrónicos para resolver las dudas en equipo.

A continuación encontrarás las consignas y tareas a desarrollar:

- **Lección 1 – Fundamentos de BDD**

 **Objetivo:** Comprender la metodología BDD y su relación con TDD.


 **Tareas a desarrollar:**

1. Leer el **Manual L1: BDD – Fundamentos de la metodología y su relación con TDD**.
2. Crear estructura del proyecto con carpetas **features** y **steps**.
3. Escribir un archivo **.feature** con 1 escenario básico (**Given**, **When**, **Then**).

→  **Métrica:**

→ 1 feature creado correctamente.

- **Lección 2 – Introducción a Cucumber**

 **Objetivo:** Configurar el entorno de Cucumber y ejecutar el primer escenario.

 **Tareas a desarrollar:**

1. Leer el **Manual L2: Características de Cucumber**.
2. Crear clase **TestRunner** con configuración básica.
3. Implementar al menos 3 pasos (**@Given**, **@When**, **@Then**) en los **StepDefinitions**.

→  **Métrica:**

→ 1 Runner y 3 pasos funcionales.

- **Lección 3 – Escritura de escenarios Gherkin**

 **Objetivo:** Redactar escenarios claros y completos en Gherkin.


 **Tareas a desarrollar:**

1. Leer el **Manual L3: El Lenguaje Gherkin**.
2. Escribir mínimo 3 escenarios relacionados con login y registro.
3. Usar **And**, **But**, narrativa y buenas prácticas.

→  **Métrica:**

→ 3 escenarios bien redactados.

- **Lección 4 – Scenario Outline**

 **Objetivo:** Reutilizar pasos para múltiples combinaciones de datos.


 **Tareas a desarrollar:**

1. Leer el **Manual L4: Escenario Outline**.
2. Crear mínimo 1 **Scenario Outline** con 3 ejemplos.
3. Parametrizar pasos usando **{string}**.

→  **Métrica:**

→ 1 outline, 3 combinaciones de datos.

- **Lección 5 – Organización con Tags**

 **Objetivo:** Filtrar y agrupar escenarios según funcionalidad.

 **Tareas a desarrollar:**

1. Leer el **Manual L5: Organización con etiquetas**.
2. Asignar tags a escenarios (**@Login**, **@CRUD**, **@SmokeTest**).
3. Ejecutar pruebas usando filtros por tag desde el **TestRunner**.

→  **Métrica:**

→ 3 tags usados y ejecutados.

- **Lección 6 – Uso de Hooks**

 **Objetivo:** Automatizar la inicialización y finalización de pruebas.

 **Tareas a desarrollar:**

1. Leer el **Manual L6: Utilización de Hooks**.
2. Crear clase de **Hooks** con métodos **@Before** y **@After**.
3. Agregar lógica común para apertura/cierre de navegador o limpieza de datos.

→  **Métrica:**

→ 2 hooks funcionales, 1 captura automática en fallos.

¿Qué vamos a validar? 🔍

1. **Correcta sintaxis Gherkin:** Que se respeten palabras clave y la estructura (Given, When, Then, etc.).
2. **Uso adecuado de Hooks:** Verificar que la inicialización y la limpieza de datos o ambientes se realice sin duplicar código en cada escenario.
3. **Aplicación de Tags:** Que los escenarios tengan etiquetas lógicas (por funcionalidad, prioridad, tipo de prueba) y se pueda ejecutar un subconjunto de pruebas selectivamente.
4. **Scenario Outline:** Que en escenarios con múltiples valores de entrada se use la sección **Examples**, manteniendo una sola definición de pasos.
5. **Reporte de ejecución:** Debe incluir el estado de cada escenario (Passed, Failed, Skipped) y la referencia a la historia de usuario validada.
6. **Trazabilidad:** Que cada escenario haga mención al requerimiento o a la historia de usuario correspondiente y que esto sea visible en el reporte final.

Referencias 🚒

Para completar este ejercicio, revisa el contenido de los 6 manuales facilitados, que brindan información detallada sobre BDD y el uso de Cucumber:

1. **Manual L1: BDD** – Fundamentos de la metodología y su relación con TDD.
2. **Manual L2: Características de Cucumber** – Introducción a la herramienta y sus ventajas.
3. **Manual L3: El Lenguaje Gherkin** – Cómo redactar escenarios de manera clara y concisa.
4. **Manual L4: Escenario Outline (Esquema del Escenario)** – Uso de Examples para pruebas centradas en datos.
5. **Manual L5: Organización de Features y Escenarios con Etiquetas** – Cómo aplicar tags para filtrar y organizar.
6. **Manual L6: Utilización de Hooks** – Manejo de @Before, @After y otras variantes para optimizar la ejecución.

Adicionalmente, se recomienda consultar la documentación oficial de Cucumber (<https://cucumber.io/docs/>) y recursos de Selenium WebDriver (<https://www.selenium.dev/>) si se va a automatizar la UI.

Recursos

- **Visual Studio Code:** Editor de texto gratuito con extensiones para Cucumber/Gherkin.
- **SQL Online:** Para quienes requieran pruebas con base de datos sencilla y no cuenten con un servidor local.
- **TidyGherkin** o extensiones similares: Para generar plantillas de Step Definitions de forma más rápida.
- **Bibliografía/Blogs:**
 - ↪ [Cucumber Docs](#)
 - ↪ [Selenium HQ](#)
 - ↪ [Blogs técnicos sobre integración de BDD y pipelines CI/CD.](#)

Entregables

1. Archivos **.feature**:
 - Deberán contener escenarios con su respectiva narrativa e incluir tags apropiados.
 - Al menos un **Scenario Outline** con su sección **Examples**.
2. **Step Definitions**:
 - Código fuente (Java, JavaScript o Python) que implemente cada paso con las anotaciones **@Given**, **@When**, **@Then**, etc.
 - Uso de Hooks (**@Before**, **@After**) para manejar inicializaciones o cierres necesarios en cada prueba.
3. **Reporte de Ejecución**:
 - Puede ser un archivo HTML, JSON o un documento complementario que muestre el resultado de cada escenario (Passed, Failed, etc.).

- Debe evidenciar la trazabilidad con la historia de usuario correspondiente (por ejemplo, “HU-001: Escenario X – Passed”).

4. Código en Repositorio (opcional):

- Si se emplea GitHub u otra plataforma, proporcionar el enlace para revisar la estructura del proyecto y la ejecución de pruebas.

Portafolio

Se sugiere incorporar este proyecto en el portafolio profesional destacando:

- **El proceso BDD** implementado.
- **Los archivos .feature** que evidencian la correcta escritura de escenarios y la colaboración con el equipo de negocio.
- **La configuración de Hooks, Tags y Scenario Outlines**, reflejando un alto nivel de conocimiento en prácticas de BDD.
- **El reporte final** mostrando la ejecución de las pruebas y la vinculación con los requerimientos o historias de usuario.

En este apartado del portafolio, se recomienda subrayar cómo la metodología BDD mejoró la comunicación y redujo malentendidos en la definición de requisitos, resaltando así el valor que la práctica aporta al ciclo de desarrollo de software.

¡Éxitos!

Nos vemos más adelante

