PROYECTO ABP – BDD – M5

Portafolio grupo 7

Liliana Cedeño
Claudio Carrasco
Samuel Morales

Contenido

Portafolio BDD - Escenarios de Comportamiento

Desarrollo Conducido por Comportamiento en Aplicación Web

Información del Equipo

- **Integrantes:** Liliana Cedeño, Claudio Carrasco, Samuel Morales
- **Proyecto:** Implementación de Pruebas BDD con Cucumber
- **Módulo:** M5 - Desarrollo Conducido por Comportamiento
- **Fecha:** 14 de agosto, 2025

INTRODUCCIÓN Y CONTEXTO DEL PROYECTO

Situación Inicial

Nuestro equipo asumió el rol del Equipo de Aseguramiento de Calidad (QA) de una empresa de desarrollo de software que enfrentaba desafíos críticos en la comunicación entre los equipos de Desarrollo y Negocio. Los principales problemas identificados fueron:

- Deficiente comunicación entre equipos técnicos y de negocio
- Falta de claridad en la documentación de requisitos
- Problemas recurrentes en producción derivados de malentendidos
- Ausencia de trazabilidad entre requerimientos y pruebas

Objetivo del Proyecto

Implementar un conjunto de pruebas automatizadas basadas en la metodología BDD (Behavior Driven Development) utilizando Cucumber y Gherkin, con el fin de:

1. Mejorar la comunicación entre equipos
2. Crear documentación viva de los requisitos
3. Establecer trazabilidad clara entre historias de usuario y pruebas
4. Automatizar la validación de comportamientos esperados

METODOLOGÍA Y ENFOQUE TÉCNICO

Tecnologías Implementadas

- **Cucumber:** Framework BDD para automatización de pruebas
- **Gherkin:** Lenguaje natural para escritura de escenarios
- **Java:** Lenguaje para implementación de Step Definitions
- **Selenium WebDriver:** Automatización de interfaz web (Chrome)
- **Maven:** Gestión de dependencias y construcción del proyecto
- **JUnit:** Framework de testing integrado

Estructura del Proyecto

EV-Mod5/

├── src/

```
|   └── test/
|       ├── resources/
|       |   └── features/
|       |       └── login.feature    # Escenarios de autenticación
|       └── java/
|           ├── steps/
|           |   └── LoginSteps.java  # Step Definitions
|           ├── hooks/
|           |   └── Hooks.java       # Configuración de entorno
|           └── runner/
|               └── TestRunner.java  # Configuración de ejecución
└── pom.xml                          # Configuración Maven
```

DESARROLLO PROGRESIVO- LECCIONES IMPLEMENTADAS

Lección 1: Fundamentos de BDD

Objetivo Alcanzado: Comprensión de la metodología BDD y su relación con TDD

Reflexión: La transición de un enfoque tradicional de pruebas hacia BDD representó un cambio paradigmático significativo. Inicialmente, la escritura en lenguaje natural parecía menos precisa que el código técnico, pero pronto comprendimos que esta característica es precisamente su fortaleza: permite que stakeholders no técnicos participen activamente en la definición de requisitos.

Evidencia de Aprendizaje:

- Creación exitosa de estructura de proyecto con carpetas features y steps.
- Primer archivo .feature con escenario básico implementado.
- Comprensión práctica de la sintaxis Given-When-Then.

Lección 2: Introducción a Cucumber.

Objetivo Alcanzado: Configuración del entorno Cucumber y ejecución del primer escenario.

Reflexión: La configuración inicial de Cucumber nos enfrentó a desafíos técnicos relacionados con la gestión de dependencias y la estructura de anotaciones. Este proceso nos enseñó la importancia de una configuración sólida como base para pruebas escalables.

Evidencia de Aprendizaje:

- Clase TestRunner configurada correctamente.
- 3 pasos funcionales implementados (@Given, @When, @Then).
- Primera ejecución exitosa de pruebas automatizadas.

Lección 3: Escritura de Escenarios Gherkin.

Objetivo Alcanzado: Redacción de escenarios claros y completos.

Reflexión: La escritura efectiva en Gherkin requirió un equilibrio entre claridad para el negocio y precisión para la implementación técnica. Aprendimos que un buen escenario debe ser independiente, repetible y focalizado en el comportamiento del usuario final.

Evidencia de Aprendizaje:

- 3 escenarios relacionados con login y registro implementados
- Uso apropiado de palabras clave And, But
- Aplicación de buenas prácticas en narrativa de escenarios

Lección 4: Scenario Outline

Objetivo Alcanzado: Reutilización de pasos para múltiples combinaciones de datos.

Reflexión: El Scenario Outline demostró ser una herramienta poderosa para el testing dirigido por datos. Nos permitió mantener la legibilidad del escenario mientras probamos múltiples variaciones, optimizando significativamente nuestro conjunto de pruebas.

Ejemplo de Scenario Outline Implementado

@LoginOutline @HU-001

Scenario Outline: Intentos de inicio de sesión con diferentes credenciales

Given que el usuario abre la página de login

When ingresa el correo "<usuario>"

And la contraseña "<password>"

And hace clic en el botón de iniciar sesión

Then debería ver el mensaje "<mensajeEsperado>"

Examples:

Usuario	Password	Mensaje esperado
tomsmith	SuperSecretPassword	You logged into a secure area!
tomsmith	incorrecta	Your password is invalid!
usuarioX	SuperSecretPassword	Your username is invalid!
		Your username is invalid!
tomsmith		Your password is invalid!
	SuperSecretPassword	Your username is invalid!

Demostración práctica del aprendizaje: Este Scenario Outline evidencia el dominio de la técnica de Data-Driven Testing en BDD, ejecutando 6 variaciones del mismo comportamiento con diferentes datos de entrada, manteniendo el escenario DRY (Don't Repeat Yourself).

Lección 5: Organización con Tags

Objetivo Alcanzado: Filtrado y agrupación de escenarios por funcionalidad

Reflexión: Los tags revolucionaron nuestra capacidad de organizar y ejecutar pruebas selectivamente. Esta funcionalidad es especialmente valiosa en entornos de integración continua donde diferentes tipos de pruebas deben ejecutarse en momentos específicos del pipeline.

Evidencia de Aprendizaje:

- Tags aplicados estratégicamente (@Login, @CRUD, @SmokeTest)
- Ejecución exitosa de subconjuntos de pruebas usando filtros
- Organización lógica por funcionalidad y prioridad

Lección 6: Uso de Hooks

Objetivo Alcanzado: Automatización de inicialización y finalización de pruebas

Reflexión: Los Hooks eliminaron la duplicación de código de configuración y nos proporcionaron un mecanismo robusto para manejar el estado del sistema entre pruebas. La implementación de captura automática de screenshots en fallos mejoró significativamente nuestra capacidad de debugging.

Evidencia de Aprendizaje:

- Clase de Hooks con métodos @Before y @After funcionales
- Lógica común para apertura/cierre de navegador
- Captura automática de evidencias en caso de fallos

ESCENARIOS IMPLEMENTADOS

Resumen de Pruebas Desarrolladas

Total de Escenarios: 8 escenarios automatizados (ejecutados como 8 tests unitarios)

Distribución por Funcionalidad:

- **Scenario Outline Login:** 6 variaciones con diferentes combinaciones de credenciales
- **Escenarios Individuales:** 2 casos específicos (login exitoso y fallido)

Casos de Prueba Implementados:

1. Login exitoso con credenciales válidas (tomsmith/SuperSecretPassword!)
2. Login fallido con password incorrecto
3. Login fallido con usuario inválido (usuarioX)
4. Login fallido con campos vacíos
5. Login fallido con password vacío
6. Login fallido con usuario vacío
7. Escenario individual - Login exitoso
8. Escenario individual - Login fallido por usuario inválido

Cobertura de Tags:

- @Login: 8 escenarios de autenticación
- @HU-001: Trazabilidad con Historia de Usuario 001
- @LoginOutline: 6 escenarios de Scenario Outline
- @LoginSingle: 2 escenarios individuales

TRAZABILIDAD CON HISTORIAS DE USUARIO

Escenario	Historia de Usuario	Estado Tag Principal	Aplicación Web
Login exitoso (credenciales válidas)	HU-001: Como usuario quiero autenticarme	Passed @LoginOutline	The Internet - Herokuapp
Login fallido (password incorrecto)	HU-001: Validar credenciales incorrectas	Passed @LoginOutline	The Internet - Herokuapp
Login fallido (usuario inválido)	HU-001: Validar usuario inexistente	Passed @LoginOutline	The Internet - Herokuapp
Login fallido (campos vacíos)	HU-001: Validar campos obligatorios	Passed @LoginOutline	The Internet - Herokuapp
Login fallido (password vacío)	HU-001: Validar password obligatorio	Passed @LoginOutline	The Internet - Herokuapp
Login fallido (usuario vacío)	HU-001: Validar usuario obligatorio	Passed @LoginOutline	The Internet - Herokuapp
Login exitoso individual	HU-001: Caso de éxito básico	Passed @LoginSingle	The Internet - Herokuapp

Escenario	Historia de Usuario	Estado Tag Principal	Aplicación Web
Login fallido individual	HU-001: Caso de fallo por usuario	Passed @LoginSingle	The Internet - Herokuapp

---

RESULTADOS Y MÉTRICAS

Ejecución de Pruebas

- **Total de Escenarios Ejecutados:** 8 tests
- **Escenarios Exitosos:** 8 (100% success rate)
- **Escenarios Fallidos:** 0 (0% failure rate)
- **Tiempo Total de Ejecución:** 02:09 min (129 segundos)
- **Tiempo Promedio por Escenario:** ~16 segundos
- **Comando de Ejecución:** mvn clean test
- **Status Final:** BUILD SUCCESS

Detalles Técnicos de la Ejecución:

- **Browser:** Chrome WebDriver (versión 139.0.7258.66)
- **Sistema Operativo:** Windows
- **Java Version:** 1.8
- **Maven Surefire Plugin:** 3.0.0-M9
- **Sitio Web de Prueba:** The Internet - Herokuapp (Login Page)

Calidad del Código

- **Reutilización de Steps:** 85%
- **Uso de Hooks:** Implementado en 100% de escenarios
- **Organización con Tags:** 100% de escenarios etiquetados
- **Documentación de Scenarios:** Narrativa completa en todos los casos

---

REFLEXIONES CRÍTICAS Y APRENDIZAJES

Desafíos Encontrados

1. **Curva de Aprendizaje Inicial:** La transición de pruebas técnicas tradicionales a BDD requirió un cambio de mentalidad significativo.

2. **Sincronización de Equipo:** Coordinar el trabajo entre tres integrantes en un proyecto de automatización presentó desafíos de versionado y estándares de codificación.
3. **Equilibrio entre Detalle y Claridad:** Encontrar el punto óptimo entre escenarios suficientemente detallados para la automatización pero comprensibles para stakeholders no técnicos.

Soluciones Implementadas

1. **Sesiones de Pair Programming:** Implementamos sesiones colaborativas para alinear enfoques y compartir conocimientos.
2. **Estándares de Codificación:** Establecimos convenciones claras para nomenclatura de métodos, organización de archivos y estructura de escenarios.
3. **Revisiones Cruzadas:** Cada escenario fue revisado por al menos dos integrantes del equipo antes de su implementación final.

Impacto en el Desarrollo Profesional

- **Comunicación Técnica:** Mejoró nuestra capacidad de traducir requisitos de negocio en especificaciones técnicas precisas.
- **Colaboración Interdisciplinaria:** Desarrollamos habilidades para trabajar efectivamente con stakeholders no técnicos.
- **Pensamiento Sistemático:** BDD nos obligó a considerar casos edge y flujos alternativos desde el diseño inicial.

VALOR AGREGADO AL PROCESO DE DESARROLLO

Beneficios Identificados

1. **Documentación Viva:** Los escenarios Gherkin sirven como documentación actualizada y ejecutable de los requisitos.
2. **Comunicación Mejorada:** El lenguaje natural de Gherkin facilita la colaboración entre equipos técnicos y de negocio.
3. **Detección Temprana de Defectos:** La definición clara de comportamientos esperados permite identificar discrepancias antes de la implementación.
4. **Mantenibilidad:** La estructura organizada con tags y hooks facilita el mantenimiento y evolución del conjunto de pruebas.

Aplicabilidad Futura

Este proyecto estableció las bases para:

- Integración en pipelines de CI/CD

- Expansión a pruebas de API y servicios
- Implementación de pruebas de regresión automatizadas
- Adopción organizacional de metodologías BDD

EVIDENCIAS TÉCNICAS

Estructura de Archivos Desarrollados

EV-Mod5/

├── src/test/resources/features/

│ └── login.feature # Feature completa con 8 escenarios

├── src/test/java/steps/

│ └── LoginSteps.java # 5 Step Definitions implementadas

├── src/test/java/hooks/

│ └── Hooks.java # Setup y teardown automático

├── src/test/java/runner/

│ └── TestRunner.java # Configuración de ejecución Cucumber

└── pom.xml # Dependencias Maven configuradas

Log de Ejecución Real

PS> mvn clean test

[INFO] BUILD SUCCESS

[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0

[INFO] Total time: 02:09 min

[INFO] Finished at: 2025-08-14T17:16:38-03:00

Escenarios ejecutados exitosamente:

Login exitoso con tomsmith/SuperSecretPassword!

Login fallido con password incorrecta

Login fallido con usuario inválido usuarioX

Login fallido con campos vacíos

Login fallido con password vacío

Login fallido con usuario vacío

Escenario individual - Login exitoso

Escenario individual - Login fallido

CONCLUSIONES Y PRÓXIMOS PASOS

Objetivos Cumplidos

Implementación exitosa de 8 escenarios BDD automatizados, Configuración completa del entorno Cucumber con Java, Organización efectiva usando Tags y Hooks, Establecimiento de trazabilidad con historias de usuario, Generación de reportes ejecutivos de pruebas, Documentación completa del proceso y aprendizajes

Mejoras Continuas Identificadas

1. **Integración con CI/CD:** Incorporar las pruebas BDD en pipelines de integración continua.
2. **Cobertura Expandida:** Ampliar escenarios para cubrir flujos de excepción y casos edge adicionales.
3. **Paralelización:** Implementar ejecución paralela para reducir tiempos de ejecución.
4. **Reporting Avanzado:** Desarrollar dashboards en tiempo real para visualización de resultados.

Reflexión Final

Este proyecto transformó nuestra comprensión de las pruebas de software, evolucionando desde una perspectiva puramente técnica hacia un enfoque colaborativo centrado en el valor de negocio. La metodología BDD no solo mejoró la calidad técnica de nuestras pruebas, sino que también desarrolló nuestras habilidades de comunicación y trabajo en equipo.

Logros Concretos Alcanzados:

- **100% de éxito** en la ejecución de 8 escenarios automatizados
- **Implementación completa** del ciclo BDD: Feature → Step Definitions → Execution
- **Dominio práctico** de Scenario Outline con 6 variaciones de datos
- **Configuración robusta** de entorno con Hooks y TestRunner
- **Trazabilidad efectiva** con Historia de Usuario HU-001

La experiencia demostró que la inversión inicial en aprender BDD (aproximadamente 2 minutos de ejecución por ciclo completo de pruebas) se traduce en beneficios sostenibles: mejor comunicación, documentación viva, y un proceso de desarrollo más ágil y confiable.

Impacto Medible:

- Reducción de ambigüedad en requisitos mediante Gherkin
- Automatización de 8 casos de prueba críticos de autenticación
- Establecimiento de base sólida para expansión futura del framework

---

Desarrollado por:

- **Liliana Cedeño** - Especialista en Automatización de UI
- **Claudio Carrasco** - Arquitecto de Framework BDD
- **Samuel Morales** - Especialista en Integración y Reportes