Un Lenguaje-Común [**Gherkin]**

A la hora de ejecutar un desarrollo, es importante alinear las expectativas de todas las partes: lo que espera el **negocio**, lo que debe construir **desarrollo**y lo que debe validar **pruebas**. Una única verdad hará mucho más fácil el éxito del producto, la herramienta que nos puede ayudar es **Gherkin**.

**Gherkin** es un meta-lenguaje muy simple, técnicamente es un **DSL**(Lenguaje de Dominio Específico) muy cercano al lenguaje natural. Usar este lenguaje es de gran ayuda para stakeholders, negocio, desarrollo y testers, ya que proporciona una forma sencilla de escribir necesidades, hacer documentación viva (que luego se va a poder ejecutar) y definir pruebas que puedan ser entendidas por todos.

La idea de Gherkin es contar cómo se comporta el software sin entrar en detalles de implementación, su objetivo es disponer de un lenguaje que sea fácil de leer tanto para desarrollo, negocio y pruebas, siendo cooperativo entre los mismos, cuando este se usa únicamente como una herramienta para automatizar pruebas y sin la participación de negocio este pierde su valor.

Para ver cómo este lenguaje nos puede ayudar en la creación de historias de usuario, documentación, pruebas unitarias y pruebas de aceptación, veamos este ejemplo:

Ejemplo1

 Yo como usuario quiero sumar números

Para evitar hacer errores tontos

Ejemplo2

  Como un ser humano

  Quiero saber la suma de dos números.

**Una Historia usando Gherkin:**

Característica: yo como usuario quiero una calculadora

  Para evitar hacer errores tontos

  Como un ser humano

  Quiero saber la suma de dos números

  Esquema del escenario: Sumar dos números

   Dado que he introducido <numero1> en la calculadora

    Y que he introducido <numero2> en la calculadora

    Y que he introducido <numero3> en la calculadora

    Cuando oprimo el <ejecutar-accion>

    Entonces el resultado debe ser <resultado> en la pantalla

Ejemplos:

    | numero1  | numero2 |ejecutar-accion | resultado |

    | 20            | 30            | 5       | sumar     | 55        |

    | 2              | 5               | 5       | resta      | -8         |

Términos Clave

   Dado  precondiciones

    Y  conector

    Cuando  acciones

    Entonces  resultado

Como podemos observar usando este lenguaje tenemos la historia, el caso o los casos de pruebas tanto unitarios como de aceptación, contamos con la documentación fácil de acoplarse a los cambios de negocio y nos genera todos los métodos que debemos de usar en la pruebas.

En términos generales, Gherkin es fácil de usar y comprender, también ayuda a reducir tiempo y a reutilizar descripciones por negocio, además nos facilita la automatización de pruebas y da una idea más clara de los flujos que se deben de automatizar para las mismas.

**EJERCICIO**

Requisitos

Yo como usuarios deseo, tener el producto de Cuenta Corriente con el siguiente comportamiento

1. **Es una cuenta corriente que tiene un crédito pre aprobado de 500 USD**

2. **La cuenta acepta depósitos sin límite de valor**

3. Puedo realizar compras con la cuenta

4. Para las compras puedo usar el saldo propio de los depósitos

5. para las compra puedo usar el saldo crédito pre aprobado

6. Cuando se realizan compras con el pre aprobado se debe validar el tope de crédito

Casos de Prueba

Cp1

Dado que se ha **creado** una cuenta corriente

Cuando **consulto el Saldo** de la cuenta corriente

Entonces puedo ver que el **valor del pre-aprobado** es de 500 USD

Cp

Dado que se ha **creado** una cuenta corriente

Y consulto el saldo de la cuenta

Y tengo un saldo a favor = 0

Y realizo un depósito por de <valor1>

Entonces puedo ver que mis saldo-a-favor final es de <saldo-a-favor>

Cp

Dado que se ha **creado** una cuenta corriente

Y tengo saldo-a-favor

Cuando realizo una compra por <valorcompra>

Entonces

Preguntas Generadas

1. **Las cuentas ya existen o deben ser creadas**
2. **Cuales son los casos**
3. **Uso del operador “O”**