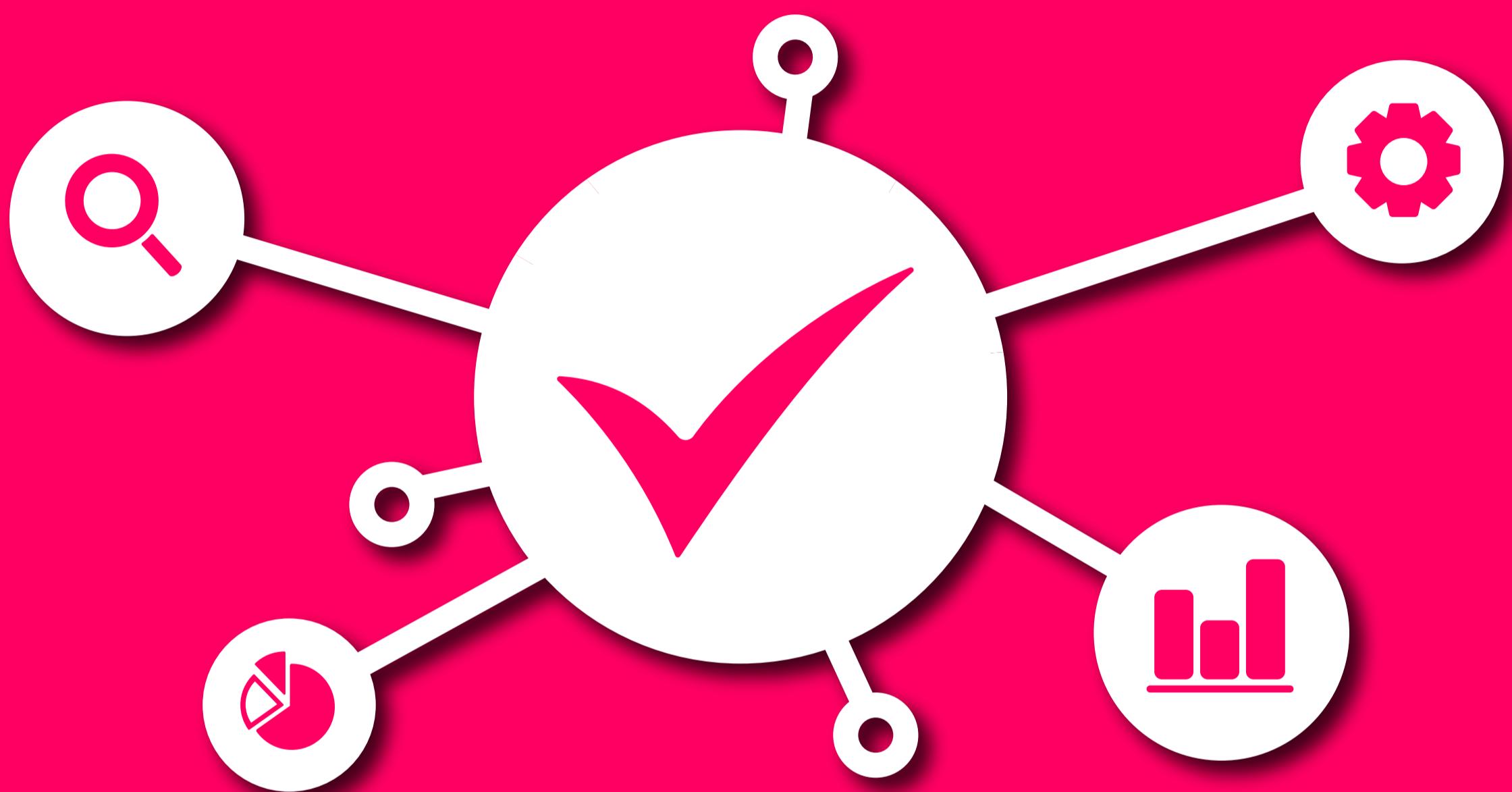




El futuro digital
es de todos

MinTIC



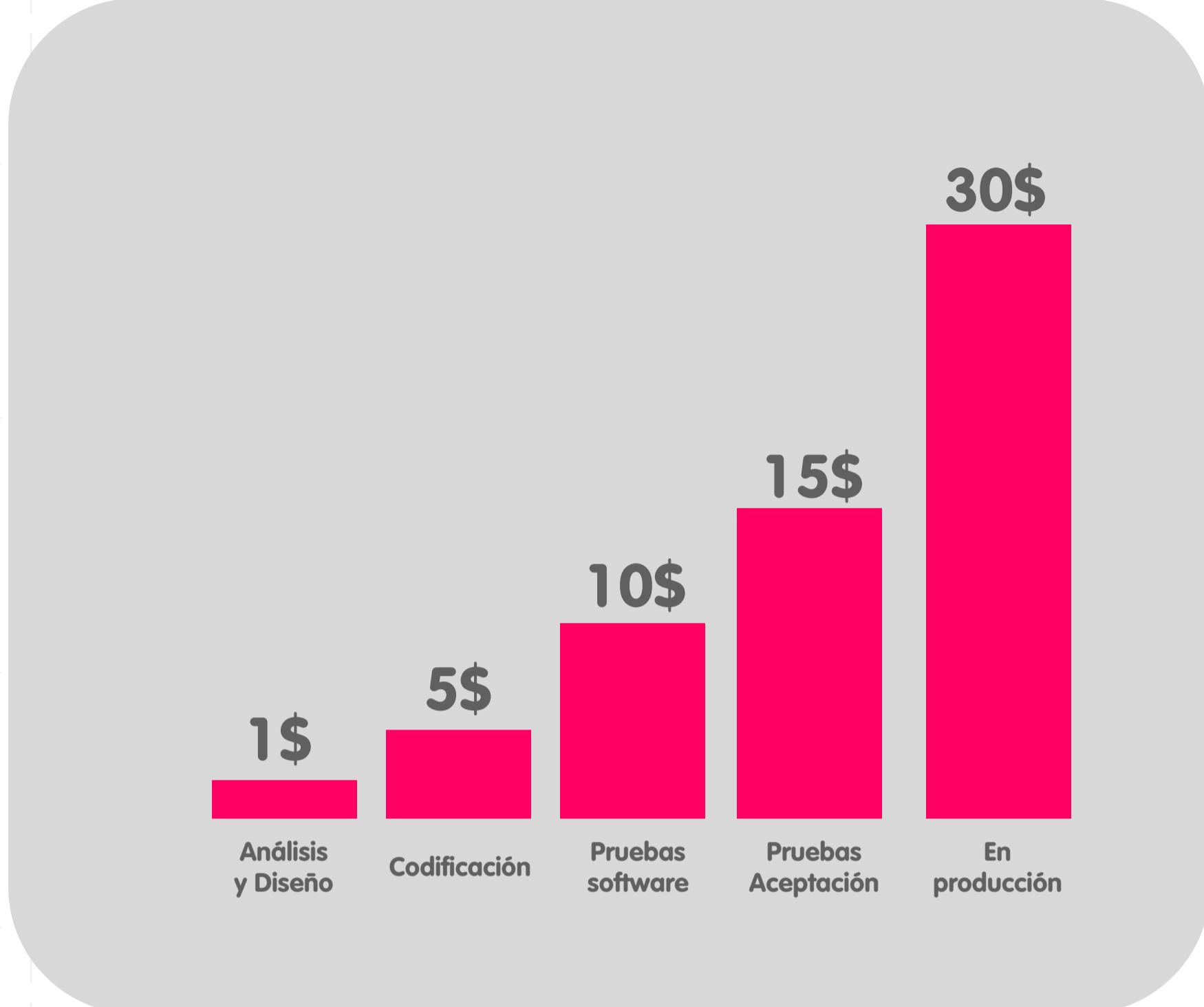
Casos de prueba

Hola:

Las aplicaciones que construimos, al igual que otros productos y servicios necesitan garantizar que cumplen aquellas cosas para las cuales fueron diseñadas, es decir, los requisitos funcionales y una de las maneras de darnos cuenta si lo hacen o no, es a través de las pruebas.

Antes de entrar directamente con el tema de pruebas analicemos el costo que tienen los errores en el desarrollo de aplicaciones y la relación que tiene ese costo con la etapa en la cual el error es identificado. Sabemos que hemos definido un proceso para el desarrollo de software y que en cualquiera de esas etapas se pueden cometer errores, pero en cualquiera de esas etapas podemos detectar esos errores si tenemos los elementos que me permiten hacerlos. Si detecto el error en esas primeras fases de análisis y diseño, eso tiene un costo debido a que amerita la creación de nuevos procesos de verificación y validación del funcionamiento de los programas.

Supongamos que allí tiene un costo dado y si los errores son detectados en el momento en que estamos codificando, esto quiere decir que el costo será cinco (5) veces más grande que en el momento de detectar el error en el análisis y diseño. Si lo detectamos en las pruebas que estamos realizando, allí ya tendrían un costo mayor porque hemos dedicado más tiempo y tengo más módulos realizados. La prueba de software la hacemos nosotros como programadores para verificar y para darnos cuenta que efectivamente lo que estamos haciendo va funcionando de acuerdo a lo que hemos desarrollado y las pruebas de aceptación son las que se hacen con los usuarios para ver si se acepta o no el programa que se está entregando. Allí el costo del error es 15 veces mayor y en caso de que se haya instalado y entregado al cliente, el costo allí será 30 veces más de lo que nos hubiera costado al inicio.



VERIFICACIÓN Y VALIDACIÓN son términos que podemos usar y en nuestro contexto es muy importante entender la diferencia.

Verificar hace referencia si estamos construyendo el producto correctamente, es decir, si estamos siguiendo el proceso debidamente y que se construyó bajo las normas y restricciones establecidas.

La **validación** se refiere si estamos construyendo el producto correcto. Aunque puede ser que sigamos todas las normas para la construcción de un barco, sin embargo, se debía construir un avión, esto sucede mucho en el desarrollo de software, es decir, **validar** lo que el usuario quiere.

Es muy importante realizar estos dos aspectos, la verificación es técnica, se hace con colegas o expertos, pero la validación es con el usuario, que efectivamente sea lo que requieren.

A continuación hay una serie de conceptos que se utilizan mucho más que **verificación** y **validación**.



Error: es una equivocación del programador.



Defecto: los errores causan defectos, como hay un error, hay una diferencia entre el valor esperado y el valor obtenido.



Falla: es el problema presentado después de desplegado el software.



Bug: es una inconsistencia de software encontrada en etapas de prueba

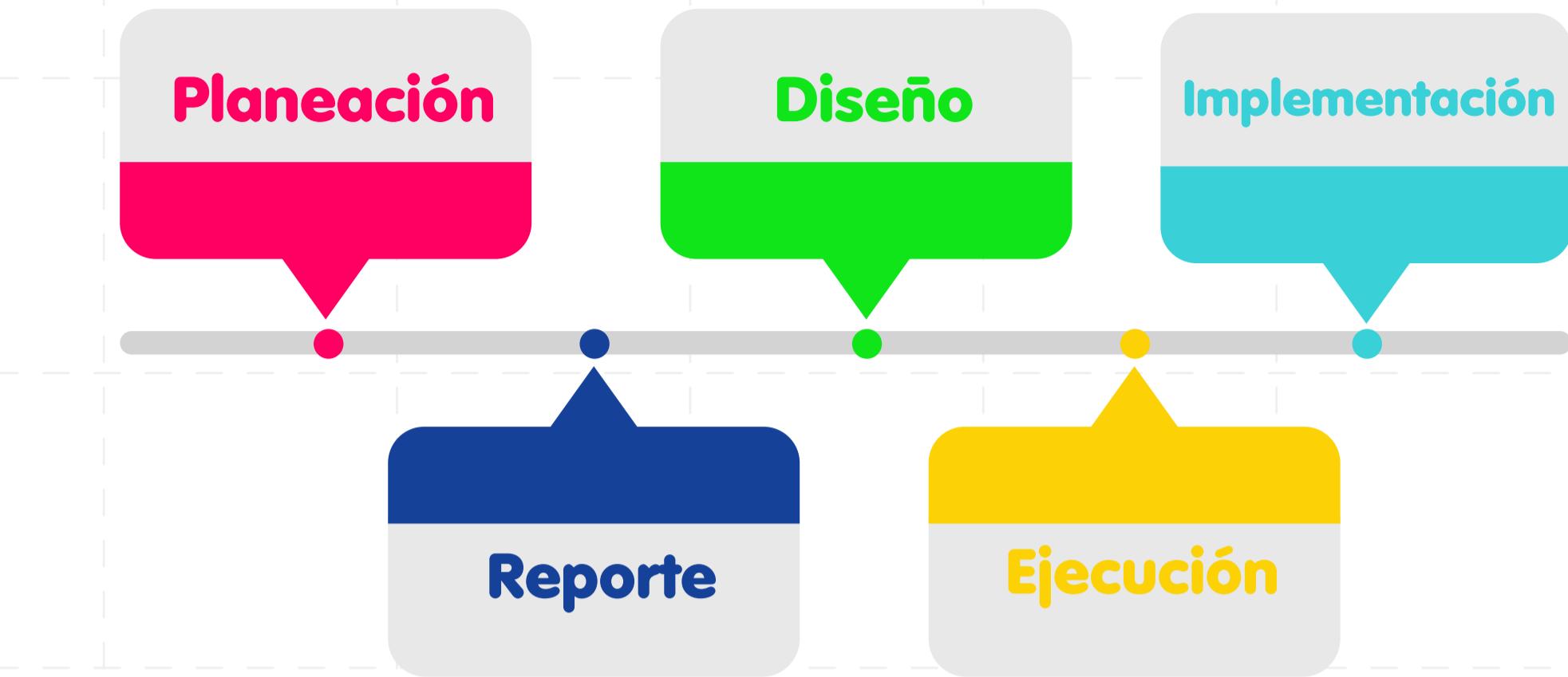
Probamos el proceso de analizar una aplicación para detectar la diferencia entre las salidas esperadas y las obtenidas que se le denominan *bugs*, que es el nombre genérico de estas diferencias. Las pruebas se realizan para detectar *bugs*.

Si estamos hablando de un proceso es debido a que tiene etapas, la primera etapa de las pruebas es la planeación de estas, definir quiénes van a hacer las pruebas, en qué fecha y plataforma.

Todas esas actividades en general hacen parte de la planeación. La segunda etapa es diseñar las pruebas con anterioridad para poderlas realizar.

La tercera etapa es la implementación, ya que las pruebas que vamos a hacer no son pruebas de usuario, sino que son pruebas automatizadas, es decir, programas que prueban otros programas.

Por último, realizamos un reporte de las inconsistencias de los bugs encontrados ya que este es el objetivo del proceso de prueba. Si no se encuentra ningún error hay dos (2) posibilidades, que en efecto no haya ningún error o las pruebas quedaron mal diseñadas.



Aquí no se encuentran todas las etapas del proceso de pruebas, pero nos interesa entender algo del diseño de prueba, que es ver cómo se implementan y ejecutan en Replit.

Un caso de pruebas es un conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo particular.



Para el diseño de casos de prueba una de las estrategias que se utiliza se llama: pruebas de caja negra. Tenemos los casos de prueba, de allí obtenemos las entradas y vemos la función como una caja negra, es decir, no sabemos cómo está implementado, sabemos que si ingresamos los datos de casos de prueba, nos genera unas salidas y lo que tenemos que hacer es comparar si esas salidas que se obtuvieron son las que se esperaban, por eso es muy importante que los casos de prueba estén muy bien escritos, verificados y validados por los usuarios, de lo contrario, las pruebas no tendrán ningún aporte.

Por ejemplo, con esta plantilla vamos a definir casos de prueba. Es importante tener una identificación nemotécnica que tenga un significado. El primer ejemplo es la muestra correcta de cómo se realiza un retiro

Identificación	CP-RS-001 - 01
Descripción	Realizar un retiro de una cuenta de ahorros
Precondiciones	- Saldo de la cuenta de ahorros: \$3.600.000
Valores de entrada	- Valor a retirar: \$3.000.000
Resultados esperados	- Saldo de la cuenta de ahorros: \$600.000

En el segundo ejemplo tenemos el mismo requisito de software, del que se espera que diga que hay un error y no tiene saldo disponible ya que hay un saldo menor del que se quiere retirar.

Identificación	CP-RS-001 - 02
Descripción	Realizar un retiro de una cuenta de ahorros
Precondiciones	- Saldo de la cuenta de ahorros: \$400.000
Valores de entrada	- Valor a retirar: \$3.000.000
Resultados esperados	- Expción: no tiene saldo disponible

En el tercer ejemplo nos solicitan un retiro de un valor negativo el cual no se puede retirar ya que debe ser mayor a cero (0).

Identificación	CP-RS-001 - 03
Descripción	Realizar un retiro de una cuenta de ahorros
Precondiciones	- Saldo de la cuenta de ahorros: \$400.000
Valores de entrada	- Valor a retirar: -\$2.000.000
Resultados esperados	- Excepción: el saldo a retirar debe ser mayor a cero

En conclusión, hemos visto el tema de pruebas y su importancia en el proceso de **verificación** y **validación** de software. Además, vimos el concepto de caja negra y casos de prueba que son muy importante para el diseño de las pruebas y ejemplos en torno a una sola funcionalidad desarrollando tres (3) casos de prueba.



Universidad de Caldas