Contents

[TESTING I 2](#_Toc118984906)

[Error: 2](#_Toc118984907)

[Defecto: 2](#_Toc118984908)

[Fallo: 2](#_Toc118984909)

[Presentación del cronograma 3](#_Toc118984910)

[Objetivo 3](#_Toc118984911)

[Consigna 4](#_Toc118984912)

[Clase 2 5](#_Toc118984913)

[Plantilla para reportar un defecto 5](#_Toc118984914)

[Diseño de la prueba 6](#_Toc118984915)

[¿Cuál es la diferencia entre severidad y prioridad? 7](#_Toc118984916)

[Técnica de los 3 amigos 8](#_Toc118984917)

[Pruebas funcionales y no funcionales 8](#_Toc118984918)

[Funcionales 8](#_Toc118984919)

[No funcionales 8](#_Toc118984920)

[Niveles de prueba 8](#_Toc118984921)

[También hay tipo de pruebas en el Testing 9](#_Toc118984922)

[Casos y defectos de prueba 10](#_Toc118984923)

[Niveles de prueba 10](#_Toc118984924)

[Técnicas de Prueba 11](#_Toc118984925)

[Caja negra 11](#_Toc118984926)

[Tecnicas de caja Negra 11](#_Toc118984927)

[Particion de equivalencia 11](#_Toc118984928)

[Analisis de Valores limites 11](#_Toc118984929)

[Tabla de Decision 12](#_Toc118984930)

[Técnicas basadas en la experiencia 12](#_Toc118984931)

[predicción de errores 13](#_Toc118984932)

[Prueba exploratoria 13](#_Toc118984933)

[Explicación de Técnicas de prueba 14](#_Toc118984934)

[¿Cual es la diferencia entre caja negra y caja blanca? 14](#_Toc118984935)

[Partición de equivalencia 14](#_Toc118984936)

[Análisis de valores limites 15](#_Toc118984937)

[Tablas de decisiones 15](#_Toc118984938)

[Transición de Estados 16](#_Toc118984939)

[Caja Blanca 16](#_Toc118984940)

[Nota de la clase donde me caí de la moto 16](#_Toc118984941)

[Pruebas Estáticas y Dinámicas 16](#_Toc118984942)

[Proceso de Revisión 17](#_Toc118984943)

[Cambiando de tema hablando sobre el proyecto integrador 17](#_Toc118984944)

[CASO DE USO 17](#_Toc118984945)

[Ambientes de trabajo 17](#_Toc118984946)

# TESTING I

Son pruebas que tenemos que desarrollar en cualquier programa o en cosas de la vida real. **PRUEBAS, partimos siempre de la suposición. No solo aplica para solo el producto si no a todos los procesos que se hacen para que el producto quede funcionando.**

Siempre cumplir con todos los estándares ya planteadas por el cliente.

**BRINDAMOS CALIDAD—SATISFACCION DEL CLIENTE.**

**Validan flujo normal de la prueba el testing positivo.**

**¿Cuáles son los roles más comunes?**

* + QA calidad de procesos
  + QC garantizar un producto final



# **Error:**

equivocación de la persona (desarrollador, QA,QC, analista)

# **Defecto:**

En el software. Incongruencia en el código

Fallo:

viene del sistema.



# Presentación del cronograma

Asistencia 90%, prácticas en play ground 80%, evaluación final mayor o igual a 7, nota final de la materia mayor o igual a 7.

1. La prueba muestra la presencia de defectos, no su ausencia

Enfoque, las pruebas están diseñadas para demostrar que existen defectos en el programa, la idea es que no se cumpla. **No nos dan la certeza que no va a existir ningún defecto.**

1. La prueba exhaustiva es imposible

Es querer hacer todas las combinaciones posibles, pero sabemos que no, tiene montones de componentes, funciones, módulos. **Nunca vamos a realizar todas las combinaciones.**

1. La prueba temprana ahorra tiempo y dinero

Nos adelantamos a los problemas muchos antes, los encontramos antes que lo encuentre el cliente.

1. Los defectos se agrupan

Ejemplo, login un modulo primario o desde antes, va a generar futuros fallos, los otros procesos se van agrupando a este.

1. Cuidado con la paradoja del pesticida

Resistencia a la misma prueba en el software.

1. La prueba se realiza de manera diferente según el contexto

La prueba tiene que ser diferente para cada escenario, tiene dependencia de la criticidad.

1. Ausencia de errores es una falacia

Siempre tiene que existir un error, si no hay prueba exhaustiva, siempre van a ver errores, nunca se puede asegurar de que no tiene errores mi programa.

Los principios nos ayudan a delimitar el trabajo a la hora de desarrollar un testing.

# Objetivo

Poner en práctica los contenidos teóricos vistos en clase 1.

# Consigna

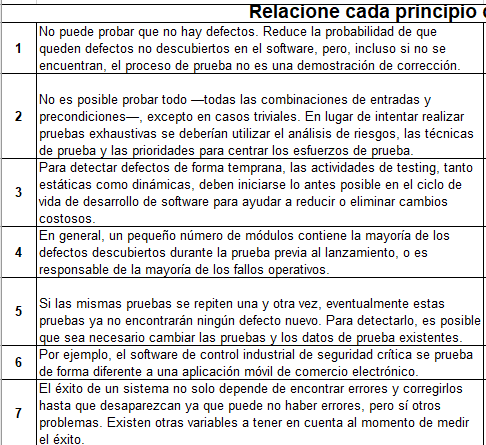
Basándote en el contenido de Playground y lo visto en clase, debatí en la mesa de trabajo y respondé lo siguiente:

1. Dentro de los miembros que forman la técnica de los tres amigos (mesa de 3 patas), ¿cuál se considera más importante? ¿Por qué?
2. Mencionar con tus palabras cuáles crees que son los principios más importantes de los 7 vistos en clase. ¿Por qué?
3. Explicar la diferencia entre verificación y validación con un ejemplo.

Solución

1. Ninguno es mas importante que otro. Todos poseen roles importantes y se complementan en el proceso de testing
2. Creemos que el mas importante es el punto 1, porque al estar desarrollado por humanos es muy probable que tenga errores debido a la magnitud de software, también porque siempre van a haber infinitas posibilidades de desarrollar algo o de testear algo, como ende también infinidad de errores no planeados.
3. Validación es que el analista haya entendido detalladamente que es lo que requiere el cliente. Historia del cliente.

Verificación, miramos de que el entregable cumpla con todos los requisitos planteados.

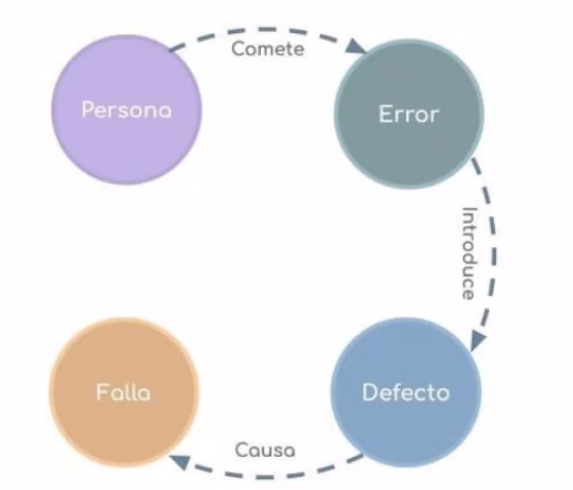


Están en orden de acuerdo a los 7 principios.

# Clase 2

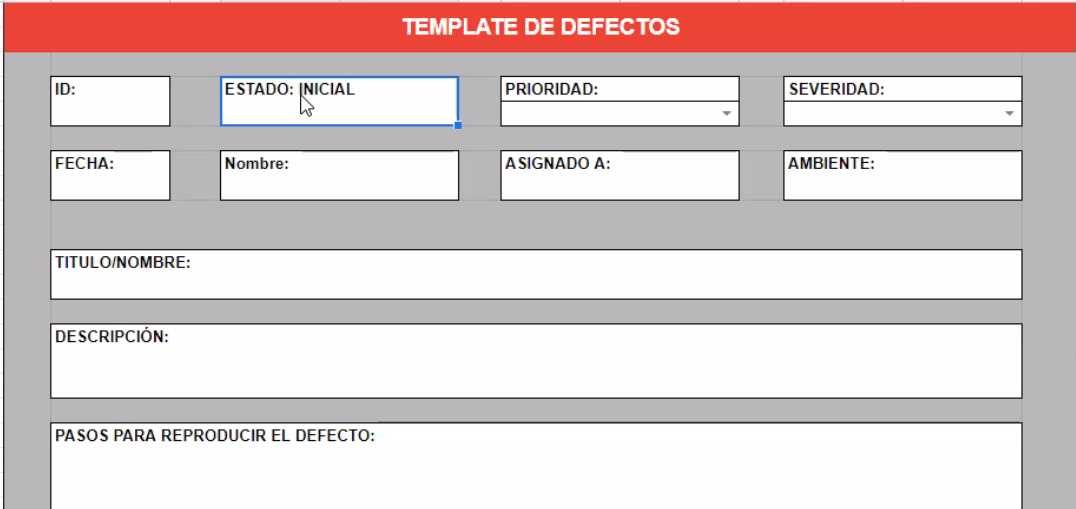
En el informe de defectos no lo relacionamos con severidad y prioridad. Ejemplo, pruebas en el modulo de login, si yo intento hacer el login con ALGO CORRECTO Y FALLA la severidad que presenta es alta, porque le impide operar el aplicativo.

Si introduzco algo incorrecto, pero no me muestra el error o bloquear la operación, tiene severidad baja, la severidad es para clasificar el defecto que se reportó, parta saber cual vamos a tomar primero.



Es importante reconocer como reportar el defecto, la idea es que lo podamos resolver más rápido y que identifiquemos que se nos está pidiendo en este.

# Plantilla para reportar un defecto



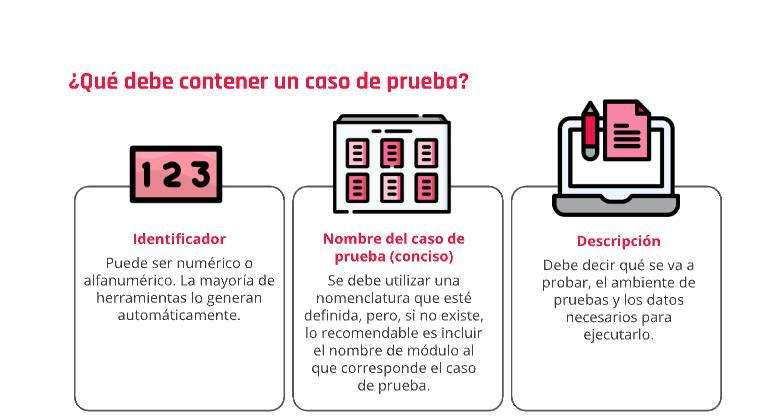
* titulo y nombre, necesitamos que sea especifico.
* Pasos para reproducir el defecto, poder describir detalladamente que es lo que se va hacer para encontrar el problema y ver como se tiene que resolver.

Mirar cómo puedo desarrollar la severidad de un defecto del programa.

# Diseño de la prueba

Es un documento escrito que proporcionar información escrita sobre y como probar.

* Deber ser simples
* El titulo debe ser fuerte
* Tener en cuenta al usuario final
* No asumir
* Asegurar la mayor cobertura posible
* Autonomía
* Evitar la repetición de casos de prueba



Diagram

Description automatically generated

# ¿Cuál es la diferencia entre severidad y prioridad?

La severidad es el impacto que tiene en el defecto, la prioridad va a ser la clasificación a la resolución en base a las actividades que tenemos que desarrollar.

# Técnica de los 3 amigos

Es la idea de la mesa de 3 patas que son necesarios en cada uno.

* Desarrollador
* Analista de negocios
* Garantía de Calidad ( QA)

# Pruebas funcionales y no funcionales

## Funcionales

Se refieren a funciones o actividades u operaciones que desarrolla el software o la página web

## No funcionales

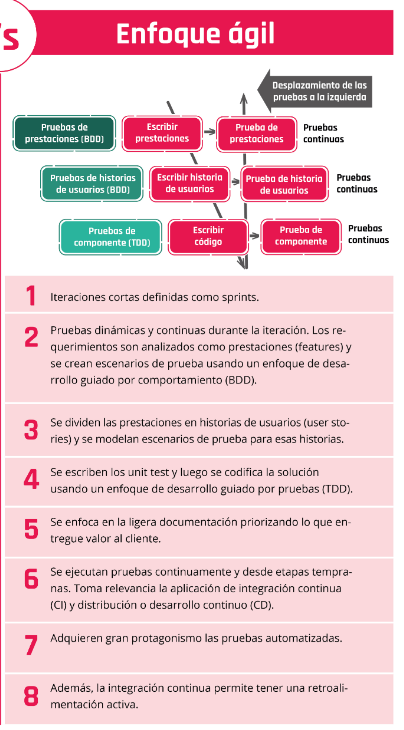
No funcionales no comprometen las actividades u operaciones y son algo más externo, en cuanto a pruebas de portabilidad, integración de componentes ETC.

## Niveles de prueba

En los niveles de prueba vamos a conocer objetivos, las bases de prueba, el objeto de prueba, los defectos y fallos característicos y los enfoque y responsabilidades especificas de cada nivel de prueba.

* Prueba unitaria o de componentes
* Prueba de integración
* Prueba de sistema
* Prueba de aceptación

Manejamos el modelo V, miramos el enfoque ágil donde hacemos una prueba en cada área por así decirlo.



## También hay tipo de pruebas en el Testing

* **Prueba funcional**: Evalúan las funciones del sistema, lo que describen al sistema.
* **No funcional**: rendimiento, optimización de tiempo, portabilidad usabilidad Y TIEMPO DE RESPUESTA
* **Estructurada**: código, flujos de trabajo, arquitectura
* **Asociadas al cambio**: RE - TEST, pruebas de nuevos requerimientos o de requerimientos en cosas ya existentes.

La idea es relacionar los niveles con las fases del proyecto.

* Pruebas de componentes: código, en bases de datos, validar una cosa por cosa.
* Nivel de integración: probar varios componentes funcionando en conjunto
* Sistema:
* Aceptación: pruebas de lo que es mas normativo, es mas responsabilidad del que pide el software

El ciclo de vida de software es el mismo que el ciclo de vida de los defectos, no

El de componentes podemos evaluar el login y el de registros, porque permite centrarnos.

# Casos y defectos de prueba

A la hora de reproducir el caso tengo que ser especifico a la hora de encontrar errores, mirar en que usuarios da, en que usuarios no da, mirar como se da el proceso de cada uno.

## Niveles de prueba

De mayor a menor, dependen del desarrollo del software

* Componentes (los más básico del sistema y hacer mas complejos después)
* Integración
* Sistemas
* Aceptación

# Técnicas de Prueba

La idea como tester es saber el tipo de pruebas que se aplican hoy en día en el mercado y pues tener un orden a la hora de desarrollarlas.

## Caja negra

En Caja negra miramos lo que entra y el resultado, no nos importa nada del código, lo que entrega y como esos prerrequisitos para que se ejecute.

Evaluamos lo que es

* Requisitos de Software
* Especificaciones
* Casos de uso
* Historias de uso

## Tecnicas de caja Negra

### Particion de equivalencia

Se dividen en particiones conocidas como clases de equivalencia, esta contiene valores que son aceptados por el sistema y también los que no son aceptados por el sistema.

Tiene una cobertura = Particiones aprobadas/particiones identificadas

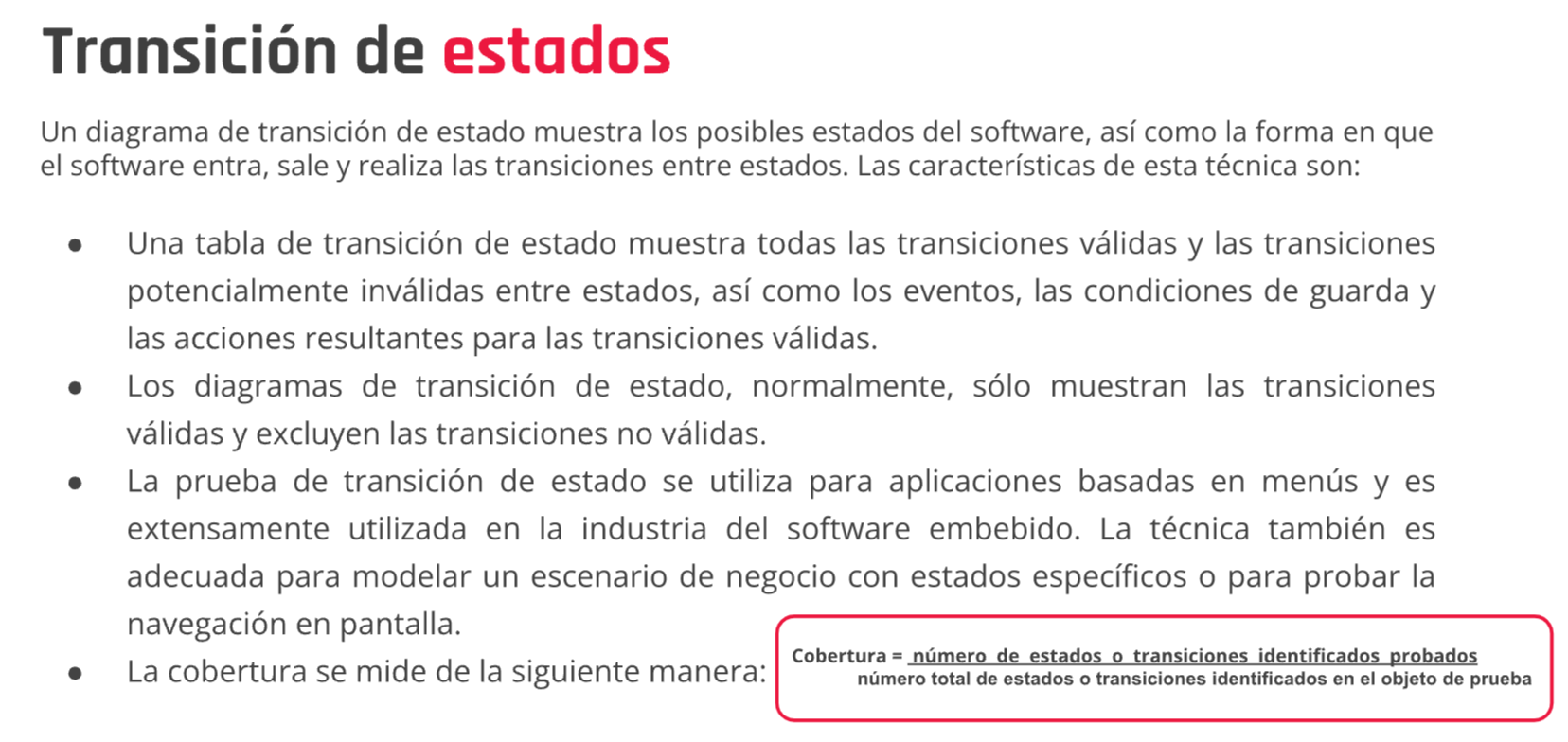
### Analisis de Valores limites

Es una extensión de partición de equivalencia



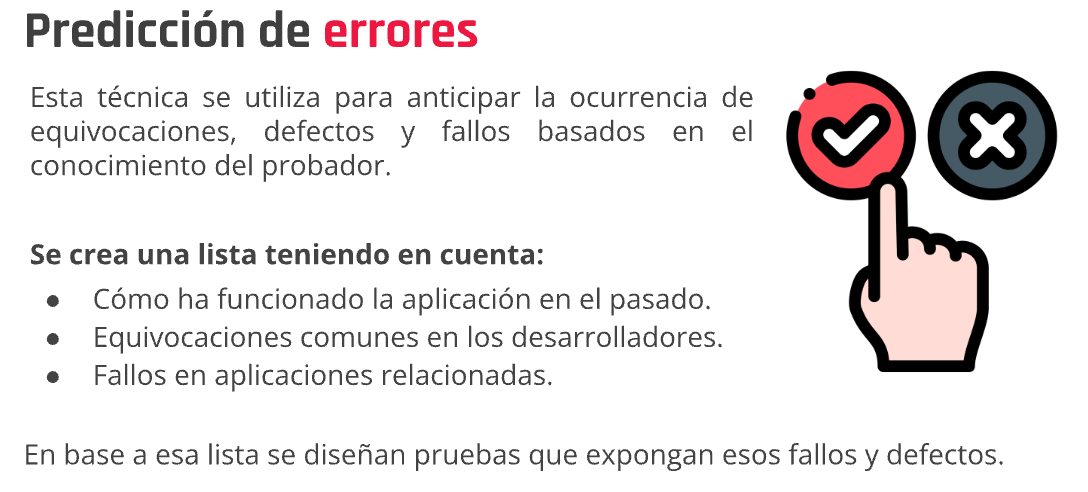
### Tabla de Decision

Se utiliza para pruebas combinatorias



# Técnicas basadas en la experiencia

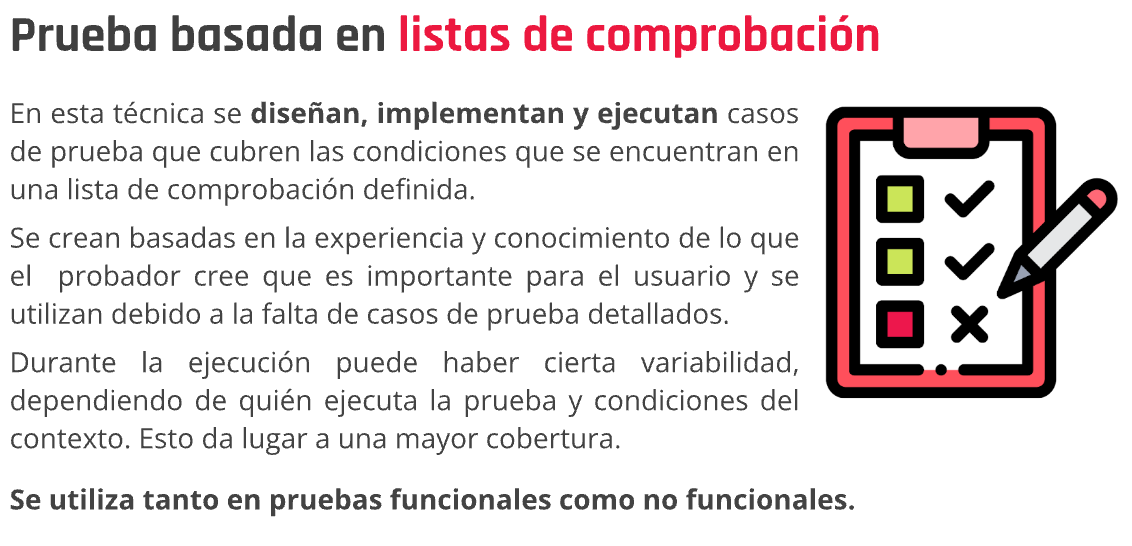
### predicción de errores



### Prueba exploratoria

Text

Description automatically generated



## Explicación de Técnicas de prueba

La cantidad de pruebas, o de escenarios que vamos a realizar o de casos hipotéticos, eso requiere demasiado tiempo. Vamos a realizar una descripción de cada una de las técnicas.

### ¿Cual es la diferencia entre caja negra y caja blanca?

En las pruebas de caja negra nos e analiza el código, solo examinamos los datos de entrada y de salida, las de caja blanca solo examinamos más el código por así decirlo.

También en caja blanca evaluamos cuales son los datos de entrada y salida. Pero mas orientado al código desarrollado por el programador.

### Partición de equivalencia

Cuando tenemos conjuntos numéricos ( Fechas, edades, generalmente cuando son muchos valores). Es una prueba que la desarrollamos por la aplicación de conjuntos, empezar a separar lo que vamos a evaluar y sus RANGOS.

Ejemplo: la validación del tipo de edad en una pagina web.

Debo de probar con valores de edad mayores a 18

**¿Cuántos valores tengo que probar hacia arriba?**

**¿Cuántos valores tengo que probar hacia abajo?**

Esta técnica nos va a devolver una participación entre dos valores, mayores de 18 y menores de 18. No es necesario que pruebe todos los valores, solo tengo que probar uno de cada uno. Solo tengo que probar en cada conjunto.

**Ejercicio nos piden desarrollar una promoción**

Los autos con antigüedad mayor a 6 años cuentan con un 15% de descuento de lo contrario, un 100% de descuento. El Black Friday no es aplicable en autos con una antigüedad mayor o igual a 12 a;os.

**Hacemos los conjuntos de nuestra promoción para aplicar la prueba**

1. Menor a 6 a;os 100%
2. Mayor a 6 hasta 12 a;os 15%
3. Mayor a 12 a;os no aplica el descuento 0a%

Tengo 3 valores donde nos van a dar 3 pruebas distintas. Si yo tomo dos valores como 2 y 9, la cobertura seria del 66%, solo cogí dos valores de 3 participaciones y cada una vale un 33%, entonces solo seria el 66% de la cobertura posible.

Recordamos que esta prueba de participación es de caja negra.

### Análisis de valores limites

Analizando el mismo ejemplo anterior, establecemos los limites de mi prueba, pongo los valores que están siguientemente. Recordamos que VALORES LIMITES es una misma extensión de la prueba de PARTICION DE EQUIVALENCIA

Timeline

Description automatically generated

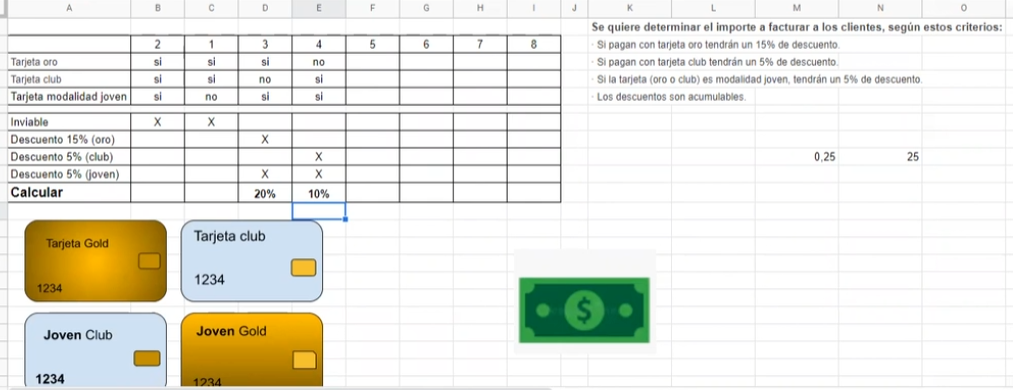
Haríamos la prueba con 1 que esta alrededor del 0, con 5 y 7 que están alrededor del 6 y al final con 11 y 13 porque están alrededor del 12.

### Tablas de decisiones

Ayuda a determinar la cantidad de decisiones, combinaciones que podemos formar. Siempre hacemos para identificar el número de reglas quedaría 2 en base n, para mirar cuantas son o cuantas columnas podemos hacer.

Ejemplo: en estos ejemplos vamos a tener un sistema de gestión de cobros para clientes de una tienda, se requiere determinar el importante a facturar a los clientes, según estos criterios:

* Si pagan con tarjeta oro tendrán un 15%
* Si pagan con tarjeta club tendrán un 5%
* Si pagan con tarjeta joven que puede ser ( oro o club ) van a tener un 5%.
* Los descuentos son acumulables, que si pago con dos tarjetas Gold, si se acumula el descuento.



### Transición de Estados

Es un diagrama, lo utilizamos mas como desarrollo para poder la cantidad de flujos que pueden suceder. Cuantas combinaciones de ciclos puedo desarrollar, cuantos me son permitidos.

## Caja Blanca

Miramos los datos de entrada y salida, pero en este caso le vamos a dar importancia a todo el software incluyendo el código.

* Código
* Estructura de software

Como objetivo principal de las técnicas de pruebas es ayudar a identificar las condiciones, los casos y los datos de prueba.

Ejecución de Casos de prueba

Lo que hacemos es llevar a cabo una serie de pasos.

Ejecución

# Nota de la clase donde me caí de la moto

Demo.testim.io

Vamos a mirar la página, se encarga en hacer viajes fuera del planeta. En la pagina vamos creando una cuenta y empezamos a validar los campos para crear la cuenta. En las técnicas, partición de equivalencia, tabla de decisiones, transición de estado y valores de limites.

## Pruebas Estáticas y Dinámicas



## Proceso de Revisión

Tenemos una documentación, siguiente paso a paso, se usaban mas en la metodología de desarrollo tradicionales. Ya que trabajamos con agiles.

Requerimientos o requisitos

* Requisitos funcionales: lo que el sistema va hacer desde un punto de vista de usuario. Nos dan una explicación básica de como quieren que se comporte el sistema.
* Requisitos No funcionales, definen condiciones de funcionamiento del sistema en el ambiente operacional.

## Cambiando de tema hablando sobre el proyecto integrador

Vamos a recibir un spring, requerimientos, los requerimientos tienen que ser específicos cuando nos dan unas condiciones en nuestro programa, necesitamos **especificidad a la hora de que nos entreguen los spring.**

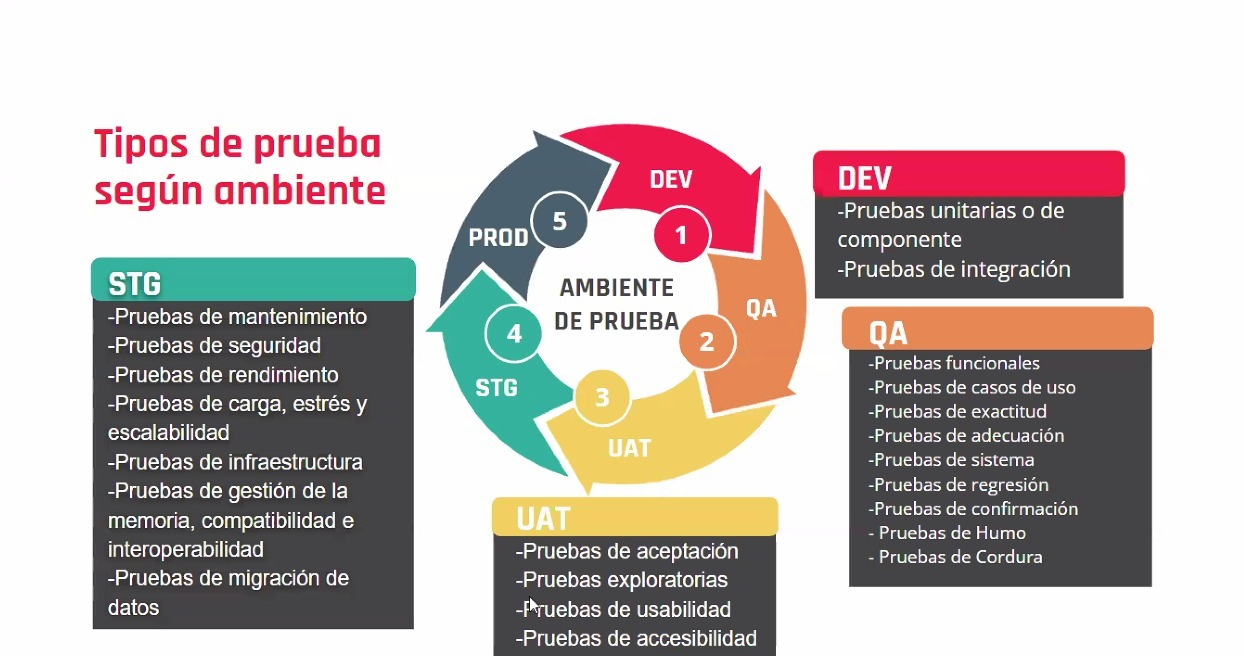
## CASO DE USO

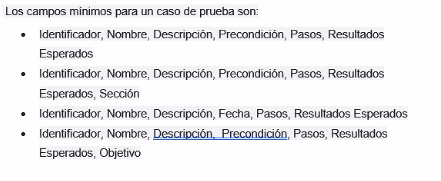
En los casos de uso tenemos que ser preguntones, en cuanto a estilos, funciones y etc.

# Ambientes de trabajo

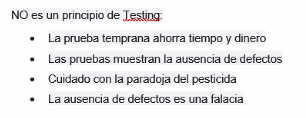
Los ambientes de trabajo es donde nosotros realizamos los cambios locos, nuevas ideas o ajustes del cliente, nada de lo que se hace ahí afecta a la aplicación.

* **Ambiente de desarrollo DEV:** el programador hace el código vía local o en la nube, pruebas iniciales y se ejecuta el programa
  + Pruebas unitarias o de componente
  + Pruebas de integración.
* **QA**: Esta ubicado en un servidor local o en la nube, ejecuta las primeras pruebas de funcionalidad
  + Pruebas funcionales
  + Pruebas de casos de uso
  + Exactitud
  + Adecuación
  + Sistema
  + Regresión
  + Confirmación
  + Humo (prueba dinámica, verifican la funcionalidad básica del sistema, son rápidas)
  + Cordura
* UAT **Prueba de aceptación de usuario:** verificación que los cambios si son los solicitados
  + Aceptación
  + Exploratorias
  + Usabilidad
  + Accesibilidad
* **STG Entorno de preproducción:** Emular el entorno de producción, probamos las actualizaciones y aseguramos que no corrompan la aplicación, antes de ser desplegadas estas.
  + Mantenimiento
  + Seguridad
  + Rendimiento
  + Carga, estrés y escalabilidad
  + Infraestructura
  + Gestión de memoria, compatibilidad e interoperabilidad
  + Migración de datos

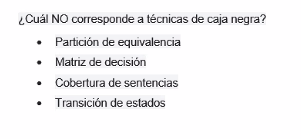




Es la 1 la respuesta.



2, la prueba muestra la ausencia de defectos



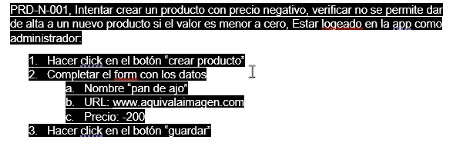
Respuesta la 3

Dise;ar un caso de prueba positivo.

Dise;ar un caso de prueba negativo

Reportar un defecto / falla en cualquiera de las funcionalidades de Digital booking

Dise;ar caso de prueba funcional que pudo haber identificado de manera temprana



***A la hora de nombrar, producto ( modulo), si es una prueba de tipo negativo o positivo, 001.***