Sistema de Gestión de Problemas

<Marco – Implementación del Testing Funcional>

Proyecto de Pruebas

Versión 1.0

Historia de Revisión

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 10/15/17 | 1.0 | Versión inicial | Emiliano Silva |
|  |  |  |  |

Tabla de Contenido

[1.Introducción 3](#_30j0zll)

[1.1Propósito 3](#_1fob9te)

[1.2Alcance 3](#_3znysh7)

[2.Estrategia de Testing 3](#_2et92p0)

[2.1Definición 3](#_tyjcwt)

[2.2Ciclo de Vida del Defecto 4](#_3dy6vkm)

[2.3Tipos de Pruebas 4](#_1t3h5sf)

[2.4Ambiente 4](#_4d34og8)

[2.5Liberaciones 4](#_2s8eyo1)

[2.6Cambios 4](#_17dp8vu)

[2.7Fases 4](#_3rdcrjn)

[2.8Herramientas 5](#_26in1rg)

[3.Resultados 5](#_lnxbz9)

# **UTEC - Modelo - Proyecto de Pruebas**

# Introducción

## Propósito

El objetivo de este plan es realizar el Testing del **Sistema Gestión de Problemas** según las especificaciones propuestas en el documento “Gestión de Problemas”, para garantizar que las diferentes funcionalidades que integran la solución cumplen con lo definido.

## Alcance

A continuación se indicarán las funcionalidades que serán probadas:

* Gestión de problemas
* Gestión de contactos
* Informes de control

A continuación se indicaran los componentes, caracteristicas y actividades que quedan fuera del alcance:

* Pruebas unitarias: se asume que todo equipo de desarrollo realiza las pruebas unitarias previo a finalizar su trabajo, por ende dichas pruebas no seran tenidas en cuenta para la estrategia de este plan.
* Las pruebas de la cosmética,
* Las prueba de diseño de las pantallas y formularios.
* Las pruebas de tipo de campo (numérico, texto, fecha, etc..) y su largo.

# Estrategia de Testing

## Definición

En primera instancia se realizara el Test Funcional, según los casos definidos en base al requerimiento. Con la ejecución se realizará el registro de defectos y la actualización de los casos de prueba.

En segundo lugar se recibirá una nueva versión donde deberá realizar el Test de Regresión de los casos de prueba que identificaron defectos y el Test Funcional.

En tercer lugar, se realizará la actualización de los defectos y de los casos de prueba.

Y por último se realizará el informe de resultados.

**Nota:**

Por cada “Build” liberado (en nuestro caso serán 2) se comenzara nuevamente desde el Test de Aceptación de Ambiente, Test de Regresión, y Test Funcional.

## Ciclo de Vida del Defecto

*Todos los* incidentes identificados durante las pruebas deberán ser registrados y gestionados por

los distintos grupos que intervienen, desde la identificación hasta su cierre pasando por la resolución.

Más adelante en el presente documento se representará el ciclo de vida, con los estados y transiciones por los que puede pasar un incidente.

De forma de hacer más comprensible dicho grafo (ciclo de vida), presentamos el siguiente escenario que

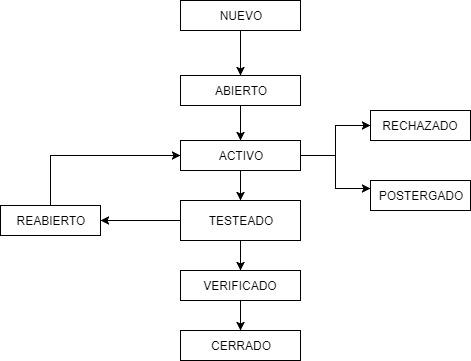
con mayor frecuencia deberá ser utilizado:

* El tester detecta un incidente a través de la ejecucion de un caso de prueba, en donde el resultado esperado no coincide con el resultado obtendo. Dicho incidente es registrado en la herramienta de gestion de incidentes(MANTIS) junto con una breve descripción y un detalle completo del mismo.El estado que se le asigna en la herramieta es “Nuevo”.
* El equipo de testing toma el incidente, verifica la a pertinencia y la prioridad del mismo , pasa el estado a “Abierto”.
* El grupo de jefes de proyecto toma el incidente y lo verifica, da su conformidad y pasa al estado de “activo”
* El grupo de analistas toma el incidente para su analisis e investigacion de su naturaleza una vez hecho esto, da su resolucion, el estado del incidente pasa a “testeado”
* El grupo de en correccion toma ese incidente, lo resuelve y cambia el estado a “verificado”
* El tester toma ese incidente, aguarda a que sea liberada la resolucion del mismo y realiza la prueba de regresion, donde al verificar que el comportamiento es el correcto pasa al estado de “cerrado”.

En cada cambio de estado del incidente se debe registrar la información pertinente.

Como escenarios diferentes podemos tener como ejemplo:

* El incidente no corresponde y se cambia el estado a “Rechazado”
* El incidente no se puede abordar en el ciclo, por lo que se cambia el estado a “Postergado”.
* El incidente pasa a estado “Reabierto” si al realizar la prueba de regresión, el error persiste
* Es necesario asignar a una persona la resolución del incidente
* Se solicita la ampliación de la información sobre la incidencia



## Tipos de Pruebas

Pruebas de Regresión: implican la re-ejecución de alguna o todas las pruebas realizadas anteriormente y relacionadas, dependiendo del alcance de la verificación.

Prueba de Modulo:Una vez que se comprobar que los modulos funcionan correctamente de forma individual, se debe comprobar que esos modulos trabajen armonicamente en conjunto, para esto se realzan la pruebas de integracion, cuyo objetivo es encontrar fallas en las interacciones entre los distintos componentes que conforman el software. El software es construido en pequeños y prueba en pequeños incrementos, siendo de esta forma mas fácil aislar y corregir los problemas.

Prueba de Integración: El objetico de las pruba de modulos es corroborar que cada modulo cumpla con el funcionamiento especificado en los requerimientos. Las funciones son probadas ingresando las entradas y examinando las salidas . Para realizar las pruebas nos basamos en los requerimientos, de donde se desprenden los casos de prueba, los que van a ser realizadas utilizando tecnicas como partición de equivalencia, análisis del valor límite, grafo causa-efecto.

## Ambiente

Debido a que la aplicación está desarrollada en Microsoft Acceso con solo instalar el paquete (en caso de no disponerlo) y ejecutando la aplicación que ha descargado de la tarea estará listo para poder ejecutarla.

## Liberaciones

La liberación de la aplicación se hará en forma completa, es decir que dispondrá de todas las funcionalidades solicitadas.

## Cambios

Una vez que el sistema sea entregado, consideraremos que es primer “Build”, “Build 1”, por cada cambio o actualización que se realice se incrementara en 1 (uno) el valor del “Build”.

Existirán dos ciclos de pruebas, donde en cada uno de ellos se ejecutarán todos los casos de prueba diseñados.

## Fases

Existirán las siguientes fases:

### Diseño de Casos de Prueba

En base a los requerimientos se van a diseñar los casos de prueba , para los cuales se va a escoger la metodología de caja negra . Metodología en la que se pasa un conjunto de datos de entrada y se ve el resultado que produce (conjunto de salida), el resultado producido es comparado con uno esperado. Para los casos de prueba diseñados se va a establecer la forma en la que se va ejecutar el caso de prueba , y se va a tomar en cuenta la prioridad de los mismos para establecer un orden de ejecución.

### Ejecución de Casos de Prueba

En base a la fase anterior, se realizara la ejecución de los casos de prueba. Esta inicia con la creación de los datos de pruebas que sean requeridos para la ejecución de los caso de prueba diseñados. La ejecución sera realizada de manera manual ,y si se detecta un fallo este deberá ser documentado y reportado haciendo uso de la herramienta Testlink y Mantis.

Cuando el defecto es reparado por el equipo de desarrollo, se deberá de realizar un re-ejecución del caso de prueba para verificar que el defecto haya sido efectivamente corregido. También se tiene que re-ejecutar todos los casos de prueba para corroborar que no se haya desencadenado otro fallo al reparar el defecto.

### Evaluación

Para la evaluación se analizaran los resultados de la ejecución de los casos de prueba, tomando en cuenta la cantidad de pasos, y de fallos (tomando en cuenta la severidad de los mismos y su incidencia funcional en el sistema); para determinar la cobertura del conjunto de componentes probados.

## Herramientas

### Testlink

Es una herramienta que permite crear y gestionar casos de pruebas, los que estarán organizados en planes de prueba. Estos permitirán a los miembros del equipo ejecutar casos de prueba y registrar los resultados. Ademas también te permitirá gestionar los resultado de los casos de prueba , contando con fácil acceso a la cantidad de casos de prueba que fallaron, la cantidad que paso, los que fueron bloqueados o cuantos quedaron sin ejecutar.

### Mantis

Es un bugTracker, es decir que es un software o aplicación que permite gestionar las incidencias que fueron generadas a partir de la ejecución de un caso de prueba . Así como generar reportes de dicho incidente y darle un seguimiento

# Resultados

*{Detalle los resultados de las pruebas realizadas al* **Sistema Gestión de Problemas}***.*

*{Incluyendo (funcionalidades probadas, defectos encontrados, la cantidad de ciclos de prueba realizados, y si la aplicación esta lista para liberarse a producción}*

La funcionalidades que fueron probadas son:

* Gestión de problemas
* Contactos y lista de contactos
* informes

Se realizaron dos ciclos de prueba con un total de 54 casos de prueba para ejecutar, para el primer ciclo se encontraron 6(seis) defectos, de los cuales 2(dos) fueron de prioridad alta, 2(dos) fueron de prioridad media y 2(dos) fueron de prioridad baja.

Para el segundo ciclo la cifra de de defectos aumento a 10(diez), de los cuales 5(cinco) permanecen de el primer ciclo, y los otros 5(cinco) son defectos nuevos. De los defectos encontrados en el primer ciclo, solo 1(uno) fue corregido.

Se encontraron que 2(dos) defectos son de prioridad alta, 5(cinco) son de prioridad media y 3(tres) son de prioridad baja. En este segundo ciclo el 40% de los defectos corresponde a la funcionalidad de informe, un 30 % corresponde a la funcionalidad de Gestión de problema, y el 30 % para lo Contactos.

Dado a que tiene 2(dos) defectos de de grado de prioridad alto, seria posible que este listo, pero seria lo ideal que no tuviese esos dos defectos de prioridad alta.