

GNC

Gestión de Nacimiento y Cría

Plan de Pruebas

Versión 1.0

**Integrantes:**

Emiliano Silva

Samuel Rodríguez

Agustín Rodríguez

Sebastián Mena

**Índice**

[Historia de Revisión 3](#_Toc529307222)

[1. Introducción 4](#_Toc529307223)

[1.1 Propósito 4](#_Toc529307224)

[1.2 Alcance 4](#_Toc529307225)

[2. Estrategia de Testing 4](#_Toc529307226)

[2.1 Definición 4](#_Toc529307227)

[2.2 Ciclo de Vida del Defecto 4](#_Toc529307228)

[2.3 Tipos de Pruebas 5](#_Toc529307229)

[2.4 Ambiente 6](#_Toc529307230)

[2.5 Liberaciones 6](#_Toc529307231)

[2.6 Cambios 6](#_Toc529307232)

[2.7 Fases 6](#_Toc529307233)

[2.7.1 Diseño de Casos de Prueba 6](#_Toc529307234)

[2.7.2 Ejecución de Casos de Prueba 6](#_Toc529307235)

[2.7.3 Evaluación 6](#_Toc529307236)

[2.8 Herramientas 6](#_Toc529307237)

[2.8.1 Testlink 6](#_Toc529307238)

[2.8.2 Mantis 7](#_Toc529307239)

[3. Resultados 7](#_Toc529307240)

# Historia de Revisión

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Descripción | Autor |
| 01/10/2018 | 1.0 | Versión inicial | GNC |
| 26/10/2018 | 1.1 | Versión mejorada | GNC |
| 10/03/2019 | 1.2 | Version mejorada | GNC |

Plan de Testing

# Introducción

## Propósito

El objetivo de este plan es realizar el Testing del **Software de gestión de cría de terneras** según las especificaciones propuestas en el documento de requerimientos, para garantizar que las diferentes funcionalidades que integran la solución cumplen con lo definido.

## Alcance

A continuación, se indicarán las funcionalidades que serán probadas:

* Login
* Alta de enfermedades
* Baja de enfermedades
* Registro de enfermedades por ternera
* Actualización de registro de enfermedades de una ternera
* Reporte Analítico de terneras enfermas por año y según la enfermedad
* Pruebas de la cosmética.
* Pruebas de diseño de las pantallas y formularios.

A continuación, se indican los componentes, características 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 serán tenidas en cuenta para la estrategia de este plan.

# Estrategia de Testing

## Definición

En primera instancia se realizará 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” liberada se comenzará nuevamente desde el Test de Aceptación de Ambiente, Test de Regresión, y Test Funcional.

## Ciclo de Vida del Defecto

Todos losincidentes 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

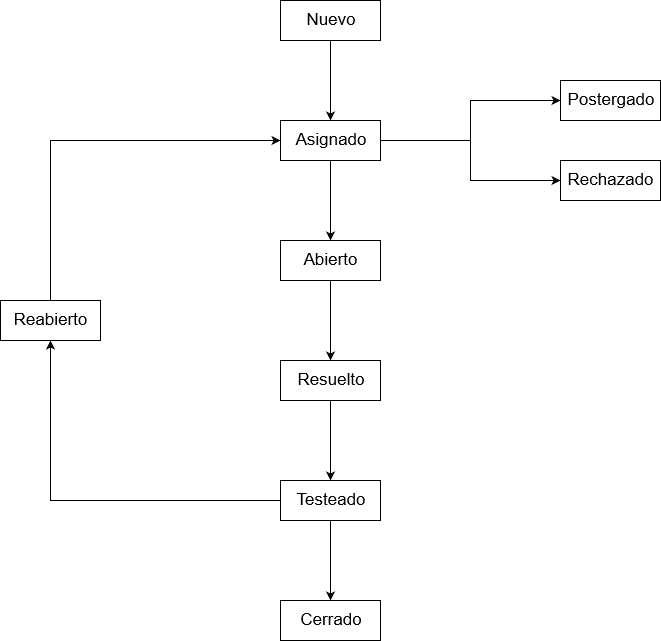
con mayor frecuencia deberá ser utilizado:

* El tester detecta un incidente a través de la ejecución de un caso de prueba, dicho incidente es registrado en MANTIS junto con una breve descripción y un detalle completo del mismo. El estado que se le asigna en la herramienta es “Nuevo”.
* Luego el equipo de desarrollo verifica si el bug es válido Si el fallo es válido, el desarrollador se asigna el bug y un estado de “asignado” se le da.
* Una vez que el desarrollador comienza a trabajar en el bug, el cambia el estado a “abierto” para indicar que él está trabajando para encontrar una solución.
* Cuando el desarrollador termina de trabajar en el bug pasa su estado a “resuelto”
* Luego el equipo de testing comienza a evaluar si fue resuelto, por lo que cambia el estado a “testeado”
* Finalmente, si el bug fue resuelto se pasa su estado a “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.



## 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 Componente: El objetivo de las pruebas de componentes es corroborar que cada uno de ellos 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 técnicas como partición de equivalencia, análisis del valor límite, grafo causa-efecto.

Prueba de Integración: Una vez que se comprueba que los componentes funcionan correctamente de forma individual, se debe comprobar que esos módulos trabajen armónicamente en conjunto, para esto se realzan las pruebas de integración, cuyo objetivo es encontrar fallas en las interacciones entre los distintos componentes que conforman el software.

## Ambiente

La aplicación es desarrollada en Eclipse y Android studio. Lo cual denota que los ambientes de prueba van a ser nuestras pc y también dispositivos Android, ya sea celulares, tablets o emulador de Android.

También es necesario tener el Wildfly bien configurado en el eclipse para poder conectar con la base de datos y que funcione la aplicación web.

## Liberaciones

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

## Cambios

Build 1: Aplicativo web con las siguientes funcionalidades:

* Login
* Agregar ternera
* Eliminar ternera
* Eliminar enfermedad
* Listado de terneras enfermas
* Listado de enfermedades

## Fases

Existirán las siguientes fases:

### Diseño de Casos de Prueba

Se diseñarán todos los casos de pruebas necesarios para cada funcionalidad de la aplicación para su futura ejecución.

### Ejecución de Casos de Prueba

Se ejecutarán todos los casos de prueba registrados y se re-ejecutaran si alguno de los casos falla.

### Evaluación

Se evaluará los resultados de la ejecución teniendo en cuenta la cantidad de casos de prueba pasados, bloqueados y fallados.

## Herramientas

### Testlink

Testlink es una herramienta Open Source que permite crear y gestionar casos de pruebas y organizarlos en planes de pruebas. Estos planes permiten a los miembros del equipo ejecutar casos de prueba y registrar los resultados dinámicamente, generar informes, mantener trazabilidad con los requerimientos, así como priorizar y asignar tareas específicas.

### Mantis

Mantis es una aplicación OpenSource hecha en php y mysql, fácil de instalar y muy flexible en su configuración. Podéis especificar un número indeterminado de estados para cada tarea (abierta, encaminada, testeada, devuelta, cerrada, reabierta…) y tantos perfiles como necesitéis (programador, tester, coordinador, visualizador…).

# Resultados

Las funcionalidades requeridas fueron programadas y cumplen con su funcionamiento. Los bugs reportados están registrados en Mantis, estos se arreglaran conforme se liberen nuevas versiones mejoradas del software en sus versiones desktop, web y mobile.

En general los resultados son buenos ya que se cumple con todas las funcionalidades requeridas.