



## **Plan de Verificación y Validación**

**Versión 1.3**

### **Historia de revisiones**

Fecha	Versión	Descripción	Autor
01/08/14	1.0	Creación del documento	Diego Melli
10/09/14	1.1	Modificación del documento	Diego Melli
15/09/14	1.2	Modificación del documento	Diego Melli
22/09/14	1.3	Modificación del documento	Diego Melli

# Contenido

<b>1. INTRODUCCIÓN</b> .....	<b>3</b>
1.1. PROPÓSITO .....	3
1.2. PUNTO DE PARTIDA .....	3
1.3. ALCANCE .....	3
1.4. IDENTIFICACIÓN DEL PROYECTO.....	4
1.5. ESTRATEGIA DE EVOLUCIÓN DEL PLAN .....	5
<b>2. REQUERIMIENTOS PARA VERIFICAR</b> .....	<b>5</b>
<b>3. ESTRATEGIA DE VERIFICACIÓN</b> .....	<b>5</b>
3.1. TIPOS DE PRUEBAS .....	5
3.1.1. <i>Prueba de Funcionalidad</i> .....	5
3.1.2. <i>Prueba de Ciclo del Negocio</i> .....	6
3.1.3. <i>Prueba de Interfase de Usuario</i> .....	6
3.1.4. <i>Prueba de Seguridad y Control de Acceso</i> .....	7
3.1.5. <i>Prueba de Documentos</i> .....	7
3.2. HERRAMIENTAS .....	8
<b>4. RECURSOS</b> .....	<b>8</b>
4.1. ROLES .....	8
4.2. SISTEMA.....	9
<b>5. HITOS DEL PROYECTO DE VERIFICACIÓN</b> .....	<b>9</b>
<b>6. ENTREGABLES</b> .....	<b>10</b>
6.1. MODELO DE CASOS DE PRUEBA .....	10
6.2. INFORMES DE VERIFICACIÓN .....	10
6.3. EVALUACIÓN DE LA VERIFICACIÓN .....	11
6.4. INFORME FINAL DE VERIFICACIÓN .....	11
<b>7. DEPENDENCIAS [OPCIONAL]</b> .....	<b>11</b>
7.1. DEPENDENCIA DE PERSONAL [OPCIONAL] .....	11
7.2. DEPENDENCIA DE SOFTWARE [OPCIONAL].....	12
7.3. DEPENDENCIA DE HARDWARE [OPCIONAL] .....	12
7.4. DEPENDENCIA DE DATOS Y BASE DE DATOS DE PRUEBA [OPCIONAL] .....	12
<b>8. RIESGOS [OPCIONAL]</b> .....	<b>12</b>
8.1. PLANIFICACIÓN [OPCIONAL].....	12
8.2. TÉCNICO [OPCIONAL] .....	12
8.3. GESTIÓN [OPCIONAL] .....	12
<b>9. APÉNDICE</b> .....	<b>13</b>
9.1. NIVELES DE GRAVEDAD DE ERROR .....	13
9.2. NIVELES DE ACEPTACIÓN PARA LO ELEMENTOS VERIFICADOS .....	13

## 1. Introducción

### 1.1. Propósito

Este Plan de Verificación para el proyecto Rexus soporta los siguientes objetivos:

- Verificación: asegurar que el software se construyó correctamente y cumple con la especificación del cliente. Se desea, mediante la realización de pruebas la detección y corrección de bugs lo antes posible reduciendo así el impacto de los mismos.
- Validación: Intentar asegurar que se construye el producto correcto, es decir el que cumple con los requerimientos del cliente.
- Calidad: Al realizar los puntos anteriores se contribuye a mejorar la calidad del producto en construcción.

Las estrategias de verificación que se usarán son:

- Revisión de código
- Gxtest para la generación de pruebas
- GxUnit para las pruebas unitarias
- Utilización de la extensión Genexus Security Scanner para identificar posibles vulnerabilidades de seguridad

### 1.2. Punto de partida

El destino de la verificación son todos los componentes desarrollados a lo largo del proyecto así como también su interacción e integración.

El objetivo del proyecto es construir una herramienta que permita gestionar requisitos y pruebas realizadas en proyectos de mediano y gran porte desarrollados usando la tecnología GeneXus.

El software a construir cuenta con 2 módulos, uno de requisitos y otro de pruebas. Ambos módulos cuentan con la integración de varias herramientas. Inicialmente ambos módulos en su totalidad serán verificados, haciendo principal énfasis sobre:

- Integración con las herramientas
- Seguimiento de las pautas de desarrollo brindadas por el cliente
- Correcta aplicación de la herramienta K2Btools
- Seguimiento de los estándares de seguridad brindados por el cliente

### 1.3. Alcance

Algunas de las características más importantes a tener en cuenta son:

- Mantenibilidad
- Usabilidad
- Seguridad

Pruebas a realizar:

#### **Unitarias:**

En una primera instancia se harán pruebas unitarias sobre todos los componentes creados.

Cada implementador estará encargado de realizar dichas pruebas a los componentes que construya, debido a:

- se aprovecha el conocimiento que estos poseen de los componentes
- tiene una curva de aprendizaje elevada al realizarse por los verificadores.

Los asistentes y responsable de verificación realizarán inspecciones de los componentes, además de que podrán participar también, asistiendo en las pruebas unitarias en caso de ser necesario.

Las pruebas serán de caja negra y blanca, utilizando la técnica de partición en clases de equivalencia y análisis de valores límites.

### **Integración:**

Muchas unidades se integran para conformar componentes compuestos, y las pruebas deben enfocarse a verificar que todos esos componentes funcionen correctamente en forma conjunta.

Además se verificará la integración con módulos creados por externos.

Las pruebas de integración serán responsabilidad del equipo de desarrollo en conjunto con el equipo de verificación.

### **Sistema:**

Finalmente se realizarán las pruebas de sistema por parte del equipo de verificación.

Dentro de las pruebas funcionales, los casos de pruebas tendrán las siguientes prioridades:

1. Casos de pruebas que fallaron en la iteración anterior
2. Casos de pruebas no ejecutados aun(nuevos)
3. Casos de prueba de Regresión
4. Casos de pruebas pendientes de la iteración anterior

Dentro de las pruebas no funcionales se verificará:

- Pruebas de seguridad
- Pruebas de mantenibilidad
- Pruebas de usabilidad

No existen requisitos de performance ni relacionados al desempeño, por lo tanto no se van a necesitar agregar pruebas de desempeño al alcance de este plan.

Riesgos: Uno de los principales riesgos que se identifican a la hora de la implementación es que todos los detalles del producto a construir no queden completamente plasmados en documentos de requisitos, casos de uso y arquitectura.

Esto llevaría a que las personas que participaron en todas las reuniones con el cliente y tienen más claro el producto a construir participen más activamente en las fases de implementación, ya sea implementando o asistiendo a los implementadores.

## **1.4. Identificación del proyecto**

Los documentos usados para elaborar el Plan de Verificación son los siguientes:

- Documento de Especificación de Requerimientos
- Transparencias de verificación y validación de la materia introducción a la ingeniería de software, y de la materia taller de Verificación de software.
- Plan de verificación y validación de otros años tomado de la Memoria Organizacional de la asignatura.

### **1.5. Estrategia de evolución del Plan**

El responsable de monitorear el Plan de Verificación y Validación es el Responsable de Verificación.

Los cambios serán realizados cuando se consideren necesarios y será comunicado a los involucrados que corresponda, por los medios de comunicación definidos en el grupo.

## **2. Requerimientos para verificar**

Los requerimientos a verificar son todos aquellos especificados en el Documento de Especificación de Requerimientos, es decir tanto aquellos requerimientos funcionales como no funcionales que hayan sido pactados con el cliente dentro del alcance del sistema.

En el transcurso del proceso se priorizarán requerimientos para verificar, que dependerán de la iteración y fase en la que se encuentra el producto.

## **3. Estrategia de Verificación**

### **3.1. Tipos de pruebas**

Pruebas de performance, de carga y volumen no serán realizadas dado que no es un requisito del sistema.

Pruebas de configuración e instalación tampoco se realizarán, dado que eso quedará a cargo del cliente.

#### **3.1.1. Prueba de Funcionalidad**

La prueba de funcionalidad se enfoca en requerimientos para verificar que se corresponden directamente a casos de usos o funciones y reglas del negocio. Los objetivos de estas pruebas son verificar la aceptación de los datos, el proceso, la recuperación y la implementación correcta de las reglas del negocio. Este tipo de prueba se basa en técnicas de caja negra, que consisten en verificar la aplicación y sus procesos interactuando con la aplicación por medio de la interfase de usuario y analizar los resultados obtenidos.

##### *3.1.1.1. Objetivo de la prueba*

Asegurar la funcionalidad apropiada del objeto de prueba, incluyendo la navegación, entrada de datos, proceso y recuperación.

##### *3.1.1.2. Técnica*

Se ejecuta cada caso de uso, flujo de caso de uso, o función usando datos válidos y no válidos, para verificar lo siguiente:

- Se obtienen los resultados esperados cuando se usan datos válidos.
- Cuando se usan datos no válidos se despliegan los mensajes de error o advertencia apropiados.
- Se aplica apropiadamente cada regla del negocio.

##### *3.1.1.3. Criterio de aceptación*

Todas las pruebas planificadas se realizaron. Todos los defectos encontrados han sido debidamente identificados.

#### *3.1.1.4. Consideraciones especiales*

Identificar o describir aquellos elementos o problemas (internos o externos) que impactaron en la implementación y ejecución de las pruebas de funcionalidad.

### **3.1.2. Prueba de Ciclo del Negocio**

Esta prueba debe simular las actividades realizadas en el proyecto en el tiempo. Se debe identificar un período, que puede ser un año, y se deben ejecutar las transacciones y actividades que ocurrirían en el período de un año. Esto incluye todos los ciclos diarios, semanales y mensuales y eventos que son sensibles a la fecha.

#### *3.1.2.1. Objetivo de la prueba*

Asegurar que la aplicación funciona de acuerdo a los requerimientos del negocio.

#### *3.1.2.2. Técnica*

La prueba debe simular ciclos de negocios realizando lo siguiente:

Las pruebas de funcionalidad se deben modificar para aumentar la cantidad de veces que se ejecuta cada función, simulando varios usuarios diferentes en un período determinado.

Todas las funciones sensibles a la fecha se deben ejecutar con fechas válidas y no válidas o períodos de tiempo válidos y no válidos.

Para cada prueba realizada verificar lo siguiente:

- Se obtienen los resultados esperados cuando se usan datos válidos.
- Cuando se usan datos no válidos se despliegan los mensajes de error o advertencia apropiados.
- Se aplica apropiadamente cada regla del negocio.

#### *3.1.2.3. Criterio de aceptación*

Todas las pruebas planificadas se realizaron. Todos los defectos encontrados han sido debidamente identificados.

#### *3.1.2.4. Consideraciones especiales*

Las fechas del sistema y eventos requieren actividades de soporte especiales. Se requieren las reglas del negocio para identificar apropiadamente los requerimientos y procedimientos a ser verificados.

### **3.1.3. Prueba de Interfase de Usuario**

Esta prueba verifica que la interfase de usuario proporcione al usuario el acceso y navegación a través de las funciones apropiada. Además asegura que los objetos presentes en la interfase de usuario se muestren como se espera y conforme a los estándares establecidos por la empresa.

#### *3.1.3.1. Objetivo de la prueba*

Verificar que: la navegación a través de los elementos que se están probando reflejen las funciones del negocio y los requerimientos, incluyendo manejo de ventanas, campos y métodos de acceso; los objetos de las ventanas y características, como menús, tamaño, posición, estado funcionen de acuerdo a los estándares, en este caso K2btools.

#### *3.1.3.2. Técnica*

Crear o modificar pruebas para cada ventana verificando la navegación y los estados de los objetos para cada ventana de la aplicación y cada objeto dentro de la ventana.

#### 3.1.3.3. *Criterio de aceptación*

Cada ventana ha sido verificada exitosamente siendo consistente con los requerimientos del cliente y el patrón k2btools.

#### 3.1.3.4. *Consideraciones especiales*

No hay por el momento.

### 3.1.4. **Prueba de Seguridad y Control de Acceso**

La Prueba de Seguridad y Control de Acceso se enfoca en dos áreas de seguridad:

- Seguridad en el ámbito de aplicación, incluyendo el acceso a los datos y a las funciones de negocios.
- Seguridad en el ámbito de sistema, incluyendo conexión, o acceso remoto al sistema.

La seguridad en el ámbito de aplicación asegura que, basado en la seguridad deseada los actores están restringidos a funciones o casos de uso específicos o limitados en los datos que están disponibles para ellos.

La seguridad en el ámbito de sistema asegura que, solo los usuarios con derecho a acceder al sistema son capaces de acceder a las aplicaciones y solo a través de los puntos de ingresos apropiados.

#### 3.1.4.1. *Objetivo de la prueba*

Seguridad en el ámbito de aplicación: Verificar que un actor pueda acceder solo a las funciones o datos para los cuales su tipo de usuario tiene permiso.

Seguridad en el ámbito de sistema: Verificar que solo los actores con acceso al sistema y a las aplicaciones, puedan acceder a ellos.

#### 3.1.4.2. *Técnica*

Seguridad en el ámbito de aplicación: Identificar y hacer una lista de cada tipo de usuario y las funciones y datos sobre las que cada tipo tiene permiso.

Crear pruebas para cada tipo de usuario y verificar cada permiso creando operaciones específicas para cada tipo de usuario.

Modificar el tipo de usuario y volver a ejecutar las pruebas para los mismos usuarios. En cada caso, verificar que las funciones o datos adicionales están correctamente disponibles o son denegados.

#### 3.1.4.3. *Criterio de aceptación*

Para cada tipo de actor conocido las funciones y datos apropiados están disponibles, y todas las operaciones funcionan como se espera y ejecutan las pruebas de Funcionalidad de la aplicación.

#### 3.1.4.4. *Consideraciones especiales*

El sistema a construir está integrado con GAM(Genexus Acces Manager), por lo que se facilitará garantizar el control y acceso al sistema.

Por otro lado se estudia la utilización de la extensión de Genexus, Security Scanner para la identificación de posibles vulnerabilidades de seguridad.

### 3.1.5. **Prueba de Documentos**

La Prueba de Documentos debe asegurar que los documentos relacionados al software que se generen en el proceso sean correctos, consistentes y entendible. Se incluyen como documentos los Materiales para Soporte al Usuario, Documentación Técnica, y todo tipo de documento que forme parte del paquete de software.

#### 3.1.5.1. *Objetivo de la prueba*

Verificar que el documento objeto de prueba sea:

- Correcto, esto es, que cumpla con el formato establecido en el proyecto.
- Consistente, esto es, que el contenido del documento sea fiel a lo que hace referencia. Si el documento es Documentación de Usuario, que la explicación de un procedimiento sea exactamente como se realiza el procedimiento en el software, si se muestran pantallas que sean las correctas.
- Entendible, esto es, que al leer el documento se entienda correctamente lo que expresa y sin ambigüedades, además que sea fácil de leer.

#### 3.1.5.2. *Técnica*

Para verificar que el documento es correcto se debe comparar con el estándar definido o con las pautas de documentación y ver que el documento cumple con ellas.

Para verificar que el documento es Consistente se debe ejecutar el programa siguiendo el documento en caso de los Materiales de Soporte al Usuario y comprobar que lo que se explica en estos documentos es exactamente lo que se ejecuta en el programa. En caso de Documentación Técnica se debe revisar el código al cual corresponde la documentación y comprobar que dicha describe el código.

Para verificar que el documento es entendible, debe comprobar que se entiende correctamente, que no tiene ambigüedades y que sea fácil de leer.

#### 3.1.5.3. *Criterio de aceptación*

El documento expresa exactamente lo que debe expresar, no hay diferencias entre lo que está escrito y el objeto de la descripción (operación de software, código de programa, decisiones técnicas) y se entiende fácilmente.

#### 3.1.5.4. *Consideraciones especiales*

No hay de momento.

### 3.2. **Herramientas**

- GxTest: Herramienta de automatización de pruebas funcionales.
- GxUnit: Herramienta para el testing unitario.
- Data Base Manager de MySQL: herramienta que se utilizará para la gestión del sistema de Base de Datos.
- Uml pacestar: Herramienta para la realización de diagramas UML
- k2btools: Herramienta que automatiza la construcción de aplicaciones Genexus.
- 

## 4. **Recursos**

En esta sección se presentan los recursos recomendados para el proyecto Rexus, sus principales responsabilidades y su conocimiento o habilidades.

### 4.1. **Roles**

En la tabla a continuación se muestra la composición de personal para el proyecto Rexus en el área Verificación del Software.

<b>Rol</b>	<b>Cantidad mínima de recursos recomendada</b>	<b>Responsabilidades</b>
Responsable de verificación	1	Identifica, prioriza e implementa los casos de prueba. <ul style="list-style-type: none"> <li>• Genera el Plan de Verificación.</li> <li>• Genera el Modelo de Prueba.</li> <li>• Evalúa el esfuerzo necesario para verificar.</li> <li>• Proporciona la dirección técnica.</li> <li>• Adquiere los recursos apropiados.</li> <li>• Proporciona informes sobre la verificación.</li> </ul>
Asistente de verificación	4	<ul style="list-style-type: none"> <li>• Ejecuta las pruebas</li> <li>• Registra los resultados de las pruebas.</li> <li>• Recuperar el software de errores.</li> <li>• Documenta los pedidos de cambio.</li> </ul>
Administrador de Base de Datos	1	<ul style="list-style-type: none"> <li>• Realiza la gestión y mantenimiento del entorno de los datos (base de datos) de prueba y los recursos.</li> <li>• Administra la base de datos de prueba.</li> </ul>

#### 4.2. Sistema

En la siguiente tabla se establecen los recursos de sistema necesarios para realizar la verificación.

<b>Recurso</b>	<b>Nombre/Tipo</b>
Servidor de base de datos	MySQL
Red o subred	Lan/local, apache tomcat
Herramientas de testing	GxTest, GxUnit, Genexus

#### 5. Hitos del proyecto de Verificación

La verificación del proyecto Rexus debe incorporar actividades de prueba para cada verificación identificada en las secciones anteriores. Se deben identificar los hitos del proyecto de verificación separados para comunicar los logros de estado de proyecto.

<b>Actividad que determina el hito</b>	<b>Esfuerzo</b>	<b>Fecha de comienzo</b>	<b>Fecha de finalización</b>
Planificar la verificación	6 horas / semanas	semana 2	semana 5
Elaborar casos de prueba	10 horas / semanas	semana 2	semana 12
Ajuste y Control de Verificación	5 horas / semana	semana 9	semana 11
Ejecutar la verificación	18 horas / semana	semana 5	semana 13
Evaluar la verificación	7 horas / semana	semana 7	semana 13

## 6. Entregables

En esta sección enumere los documentos, herramientas e informes que se crearán, por quien, para quién y cuándo serán liberados.

Para cada entregable deberá indicar las fechas en que son liberadas todas las versiones del mismo.

### 6.1. Modelo de Casos de Prueba

Documento	<b>Modelo de Casos de Prueba</b>
Creado por	El Responsable de verificación, Diego Melli.
Para quien	Es la guía para realizar las pruebas del sistema y lo usarán los Asistentes de verificación, Desarrolladores y el Responsable de verificación cuando se ejecuten las pruebas del sistema.
Fecha de liberación	Será liberado el 22/09/14

### 6.2. Informes de Verificación

Documento	Se genera un documento <b>Informe de Verificación Unitaria</b> por cada prueba unitaria que se realice al sistema.
Creado por	Las personas que ejecutan las pruebas.
Para quien	Es el retorno para los implementadores de la tarea de verificación, que detalla los errores encontrados para que puedan ser corregidos.
Fecha de liberación	Será liberado luego de cada verificación unitaria.

Documento	Se genera un documento <b>Informe Consolidación</b> por cada consolidación que se realice al sistema.
Creado por	Las personas que ejecutan las pruebas.
Para quien	Es el retorno para los implementadores de la tarea de consolidación, que detalla los errores encontrados para que puedan ser corregidos.
Fecha de liberación	Será liberado luego de cada consolidación.

Documento	Se genera un documento <b>Informe de Verificación de Integración</b> por cada prueba de integración que se realice al sistema.
Creado por	Las personas que ejecutan las pruebas.
Para quien	Es el retorno para los implementadores de la tarea de verificación, que detalla los errores encontrados para que puedan ser corregidos.
Fecha de liberación	Será liberado luego de cada verificación de

	integración.
Documento	Se genera un documento <b>Informe de Verificación de Sistema</b> por cada prueba de sistema que se realice.
Creado por	Las personas que ejecutan las pruebas.
Para quien	Es el retorno para los implementadores de la tarea de verificación, que detalla los errores encontrados para que puedan ser corregidos.
Fecha de liberación	Será liberado luego de cada verificación de sistema.

### 6.3. Evaluación de la verificación

Documento	Se genera un documento <b>Evaluación de la verificación</b> por cada prueba que se realice al sistema. Este documento contiene las fallas encontradas en el sistema, la cobertura de la verificación realizada y el estado del sistema.
Creado por	El Responsable de verificación, que toma como fuente de su trabajo los Informes de verificación.
Para quien	Es el resumen de la tarea de verificación y es el retorno para todo el equipo de trabajo del estado del sistema.
Fecha de liberación	Será liberado luego de cada verificación, unitaria, de integración y de sistema.

### 6.4. Informe final de verificación

Documento	El documento <b>Informe final de verificación</b> es el resumen de la verificación final del sistema antes de que sea liberado al entorno del usuario.
Creado por	El Responsable de verificación, que toma como fuente de su trabajo los Informes de verificación.
Para quien	Indica el estado del sistema.
Fecha de liberación	Será liberado luego de la verificación final del sistema.

## 7. Dependencias [opcional]

[En esta sección se detallan las dependencias, si existen, de las actividades de verificación respecto a otros elementos del sistema.]

### 7.1. Dependencia de personal [opcional]

[En esta sección se detallan las necesidades de personal para el equipo de verificación, la cantidad y habilidades.]

### 7.2. Dependencia de software [opcional]

[En esta sección se detallan los requisitos que debe cumplir el software a ser verificado para que se realice la verificación. Por ejemplo, que debe tener una verificación previa por parte del implementador, que debe estar en fecha disponible para verificar.]

### 7.3. Dependencia de hardware [opcional]

[En esta sección se detallan las necesidades de disponibilidad de hardware para realizar las tareas de verificación. Por ejemplo, horario, cantidad de horas que

debe estar disponible para que las tareas de verificación se puedan realizar dentro del cronograma establecido.]

#### **7.4. Dependencia de datos y base de datos de prueba [opcional]**

[En esta sección se detallan las necesidades de disponibilidad de dato y de la base de datos para realizar las tareas de verificación. Por ejemplo, conjuntos de datos necesarios para la verificación, horarios de acceso a la base de datos que permitan que las tareas de verificación se puedan realizar dentro del cronograma establecido.]

### **8. Riesgos [opcional]**

[En esta sección se detallan los riesgos detectados que puedan afectar la normal realización de las tareas de verificación.]

#### **8.1. Planificación [opcional]**

[En esta sección se plantean los riesgos relativos a la planificación. Por ejemplo, si una cronograma es muy ajustado un pequeño retraso en la liberación del software para verificar atrasa la verificación y por consiguiente las actividades que dependen de esta, provocando un retraso en el cronograma de todo el proyecto.]

#### **8.2. Técnico [opcional]**

[En esta sección se plantean los riesgos técnicos que afectan a la verificación. Por ejemplo, si existe un sistema anterior se deberá hacer una verificación en paralelo con el anterior en lugar de liberar la versión en un punto de trabajo a modo de prueba del sistema.]

#### **8.3. Gestión [opcional]**

[En esta sección se plantea como mediante la gestión se pueden mitigar los riesgos en las tareas de verificación.]

## 9. Apéndice

### 9.1. Niveles de gravedad de error

En muchas actividades del proceso de verificación se deben clasificar los errores según su nivel de gravedad. Se asigna un nivel de gravedad a los errores para poder capturar de alguna manera su impacto en el sistema. Además para poder evaluar la verificación y el sistema.

A continuación se da una sugerencia de cuatro niveles diferentes de gravedad de error:

- **Catastrófico:** un error cuya presencia impide el uso del sistema.
- **Crítico:** un error cuya presencia causa la pérdida de una funcionalidad crítica del sistema. Si no se corrige el sistema no satisfará las necesidades del cliente.
- **Marginal:** un error que causa un daño menor, produciendo pérdida de efectividad, pérdida de disponibilidad o degradación de una funcionalidad que no se realiza fácilmente de otra manera.
- **Menor:** un error que no causa perjuicio al sistema, pero que requiere mantenimiento o reparación. No causa pérdida de funcionalidades que no se puedan realizar de otra manera.

### 9.2. Niveles de aceptación para lo elementos verificados

Se debe establecer un nivel de aceptación para los elementos verificados para poder establecer el estado en el que se encuentra el proyecto.

En esta sección defina niveles de aceptación y los criterios de pertenencia a cada nivel.

Como ejemplo de niveles de aceptación:

- **No aprobado:** el elemento verificado tiene errores catastróficos (uno o varios) que impiden su uso o tiene errores críticos (uno o varios) que hacen que el elemento verificado no sea confiable. El usuario no puede depender de él para realizar el trabajo.
- **Aprobado con Observaciones:** el elemento verificado no tiene errores catastróficos, ni errores críticos, pero tiene errores marginales (uno o varios) que hacen que el elemento de software se degrade en algunas situaciones.
- **Aprobado:** el elemento verificado no tiene errores o tiene errores menores que no afectan el normal funcionamiento del elemento.