

GVA

Grupo 2

Semana 4

Plan de verificación y validación

Versión 2.0

Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|------------|---------|--|---------------------|
| 29/08/2014 | 1.0 | Creación del documento | Christopher Quincke |
| 12/09/2014 | 2.0 | Se agregan pruebas de performance y se quitan aquellos tipos de pruebas que no se realizarán | Christopher Quincke |
| 13/09/2014 | 2.0 | Validación SQA | Alejandro Casco |
| | | | |

Contenido

| | |
|---|-----------|
| 1.INTRODUCCIÓN..... | 3 |
| 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 |
| 2.REQUISITOS PARA VERIFICAR..... | 5 |
| 3.ESTRATEGIA DE VERIFICACIÓN | 5 |
| 3.1.TIPOS DE PRUEBAS | 6 |
| 3.1.1.Prueba de integridad de los datos y la base de datos..... | 6 |
| 3.1.2.Prueba de Funcionalidad | 6 |
| 3.1.3.Prueba de Ciclo del Negocio..... | 7 |
| 3.1.4.Prueba de Interfase de Usuario..... | 7 |
| 3.1.5.Prueba de Performance..... | 8 |
| 3.1.6.Prueba de Carga | 9 |
| 3.1.7.Prueba de Esfuerzo (stress, competencia por recursos, bajos recursos) | 10 |
| 3.1.8.Prueba de Volumen | 11 |
| 3.1.9.Prueba de Seguridad y Control de Acceso | 11 |
| 3.1.10.Prueba de Fallas y Recuperación..... | 12 |
| 3.1.11.Prueba de Configuración | 13 |
| 3.1.12.Prueba de Instalación..... | 14 |
| 3.1.13.Prueba de Documentos..... | 15 |
| 3.2.HERRAMIENTAS | 15 |
| 4.RECURSOS | 16 |
| 4.1.ROLES | 16 |
| 4.2.SISTEMA..... | 16 |
| 5.HITOS DEL PROYECTO DE VERIFICACIÓN | 17 |
| 6.ENTREGABLES | 17 |
| 6.1.MODELO DE CASOS DE PRUEBA | 17 |
| 6.2.INFORMES DE VERIFICACIÓN..... | 17 |
| 6.3.EVALUACIÓN DE LA VERIFICACIÓN | 18 |
| 6.4.INFORME FINAL DE VERIFICACIÓN..... | 18 |
| 7.DEPENDENCIAS | 19 |
| 7.1.DEPENDENCIA DE PERSONAL | 19 |
| 8.RIESGOS | 19 |
| EN ESTA SECCIÓN SE DETALLAN LOS RIESGOS DETECTADOS QUE PUEDAN AFECTAR LA NORMAL REALIZACIÓN DE LAS TAREAS DE VERIFICACIÓN..... | 19 |
| 8.1.PLANIFICACIÓN | 19 |
| 9.APÉNDICE | 20 |
| 9.1.NIVELES DE GRAVEDAD DE ERROR | 20 |
| 9.2.NIVELES DE ACEPTACIÓN PARA LOS ELEMENTOS VERIFICADOS..... | 20 |

1. Introducción

1.1. Propósito

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

- Identificar la información de proyecto existente y los componentes de software que deben ser verificados.
- Enumerar los requerimientos recomendados para verificar.
- Recomendar y describir las estrategias de verificación que serán usadas.
- Identificar los recursos necesarios y proporcionar una estimación de esfuerzo para realizar la verificación.
- Enumerar los entregables del proyecto de verificación.

1.2. Punto de partida

Los objetivos de la validación y verificación son:

- Detectar y corregir los defectos tan pronto como sea posible en el ciclo de vida del software.
- Disminuir los riesgos, las desviaciones sobre los presupuestos y sobre el programa de tiempos.
- Mejorar la calidad y fiabilidad del software.
- Mejorar la visibilidad de la gestión del proceso de desarrollo.
- Valorar rápidamente los cambios propuestos y sus consecuencias.

El proyecto VGA consiste de una aplicación web cliente-servidor que tiene como fin mantener la trazabilidad de activos de la empresa Sonda. La aplicación web debe poder ser utilizada desde dispositivos móviles. Entre las funcionalidades más importantes de la aplicación se encuentran: alta de activos, alta de eventos asociados a activos y usuarios del sistema, consulta sobre el historial de eventos asociado a un activo, importación y exportación de datos desde y hacia planillas excel, notificaciones sobre eventos programados, geolocalización de activos.

1.3. Alcance

El plan de verificación y validación que se presenta a través de este documento, aplicará a todos los componentes de software y a todos los documentos que formen parte del producto del proyecto.

El proceso de verificación se basa en cuatro fases. Las mismas serán realizadas en el orden en el que se presentan a continuación.

Testing unitario

Se verifican los componentes del software de forma unitaria. Esta tarea será llevada a cabo por el implementador encargado de desarrollar el componente, de forma de aprovechar el conocimiento que tenga sobre el modulo desarrollado. Al efectivizar las pruebas correspondientes, el implementador deberá escribir un reporte de las pruebas realizadas indicando que pruebas hizo y si alguna falló. Este reporte deberá adjuntarse en el mantis correspondiente al requisito que resuelve. Para el reporte se le proveerá al implementador una plantilla.

Testing de integración

A partir de los componentes previamente verificados, se procederá a probar la interacción entre estos teniendo en cuenta los requisitos a cumplir.

Se verifica que el sistema cumpla con los requisitos descritos en la documentación correspondiente y que cuentan con la aprobación del cliente. Esta etapa de la verificación será llevada a cabo por el equipo de verificación. Los casos de prueba estarán basados en los casos de uso y la modalidad de las pruebas será caja negra.

Testing de Aceptación

En esta etapa el cliente o el grupo de usuarios finales verifican que el sistema cumple con los requisitos acordados previamente.

Un requisito no funcional solicitado por el cliente es tener la mayor documentación posible acerca de la verificación realizada, por lo tanto es necesario que quienes verifiquen se tomen el tiempo de documentar que pruebas hicieron.

Algunos de los riesgos que se pueden presentar en el proyecto:

- Agregar u omitir requisitos. Esto dependerá de una buena validación de los requisitos con el cliente y de una buena negociación del alcance.
- La disponibilidad de los recursos a verificar. Un atraso en el cronograma de implementación influirá directamente en el tiempo disponible para realizar las pruebas. Esta posible demora deberá ser mitigada mediante la preparación y disponibilidad de las pruebas, para poder atacar la verificación apenas sea posible.
- Dificultad de aprendizaje de las herramientas disponibles para realizar las pruebas. Las herramientas serán Mantis Bug Tracker.
- Imposibilidad de verificar algún componente. Dado que se utilizarán herramientas y tecnologías desconocidas, es importante asegurar de antemano que todos los componentes sean verificables. Lo realizado en la fase inicial del proyecto es esencial para dicha validación.

1.4. Identificación del proyecto

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

- Plantilla del Plan de Verificación y Validación especificada en el Modelo de Proceso Modularizado Unificado.
- Documento de Descripción del Proyecto y de Especificación de Requerimientos.
- Planes de Verificación y Validación de Proyectos anteriores tomados de la Memoria Organizacional de la asignatura.
- Diapositivas de Introducción a la Ingeniería de Software acerca de Verificación y Validación.

1.5. Estrategia de evolución del Plan

El responsable del monitoreo del plan de verificación y validación será el encargado de verificación. Durante las dos primeras fases deberá planificar la verificación que se realizará a lo largo del proyecto. El plan podrá ser inestable durante las dos primeras fases del proyecto considerando los cambios en los requisitos que puedan surgir. Finalizada la segunda etapa se tendrá un plan estable y acorde a los requisitos del cliente.

Durante las dos primeras semanas se estima que se reverá el plan semanalmente dado que se estará definiendo el alcance con el cliente.

Posteriormente el plan será revisado cada dos semanas aproximadamente, teniendo en cuenta que ya se tendrán requisitos más estables.

Los cambios en el plan pueden ser surgir como consecuencia de sugerencias de cualquier integrante del equipo, pero la aprobación de los cambios la realizará el equipo de verificación. Los cambios serán informados al equipo mediante los medios de comunicación establecidos al inicio del proyecto por el equipo.

2. Requisitos para verificar

Los requisitos a verificar son todos los que se especifiquen en el documento de especificación de requisitos, funcionales y no funcionales que hayan sido acordados dentro del alcance del proyecto con el cliente. También deberán ser verificados todos los casos de uso especificados en el documento de modelado de casos de uso. Los requisitos a verificar serán priorizados dependiendo de la fase y la iteración en la que se encuentre el producto.

3. Estrategia de Verificación

Como se comentó anteriormente en el punto 1.3, la verificación constará de cuatro fases. En la primera de ellas, el testing unitario, cada desarrollador deberá diseñar casos de prueba y ejecutarlos. Para ello, dependiendo del modulo a probar podrá utilizar la herramienta de automatización de verificación NUnit. Es de suma importancia que en esta etapa se documente cuales fueron las pruebas realizadas y si alguna falló. El encargado de verificación tendrá que, en base a la documentación generada por el implementador, aprobar o desaprobado la verificación realizada. En caso de que la considere insuficiente podrá diseñar nuevas pruebas y solicitarle al implementador que las ejecute.

Para las pruebas de integración anteriores a una liberación del sistema se define el siguiente ciclo de vida:

Diseño de pruebas

- Realizadas por el encargado de verificación con la colaboración de todo el equipo de verificación.
- Se diseñan en paralelo mientras los implementadores desarrollan los componentes de software que les hayan sido asignados.

Preparación del ambiente

- Se prepara el ambiente para ejecutar las pruebas diseñadas previamente.

- Realizada por el SCM, verificadores y de ser necesario especialistas técnicos.

Ejecución de las pruebas

- Realizada por el equipo de verificadores.
- Resultados posibles: "Aprobado", "No Aprobado", "Aprobado con observaciones".

3.1. Tipos de pruebas

3.1.1. Prueba de integridad de los datos y la base de datos

3.1.1.1. Objetivo de la prueba

Asegurar que los métodos y procesos de acceso a la base de datos funcionan correctamente y sin corromper datos. Es muy importante la integridad transaccional en cualquier tipo de sistema y en particular en este caso fue solicitado explícitamente por el cliente.

3.1.1.2. Técnica

Invoque cada método o proceso de acceso a la base de datos con datos válidos y no válidos. Inspeccione la base de datos para asegurarse de que se han guardado los datos correctos, que todos los eventos de la base de datos ocurrieron correctamente, o repase los datos devueltos para asegurar que se recuperaron datos correctos por la vía correcta.

3.1.1.3. Criterio de aceptación

Todos los métodos y procesos de acceso a la base de datos funcionan como fueron diseñados y sin datos corruptos.

3.1.1.4. Consideraciones especiales

La prueba requiere un entorno de administración de DBMS o controladores para ingresar o modificar información directamente en la base de datos.

Los procesos deben ser invocados manualmente.

Se deben usar bases de datos pequeñas para aumentar la facilidad de inspección de los datos para verificar que no sucedan eventos no aceptables.

Dado que algunas de estas pruebas serán realizadas en modalidad caja negra con un servidor "deployado" en un servidor remoto se deberá planificar con los especialistas técnicos como se accederá a la base de datos del servidor.

3.1.2. 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 interfaz de usuario y analizar los resultados obtenidos.

3.1.2.1. Objetivo de la prueba

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

3.1.2.2. Técnica

Ejecute 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.

3.1.2.3. Criterio de aceptación

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

3.1.2.4. Consideraciones especiales

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

3.1.3. 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.

3.1.3.1. Objetivo de la prueba

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

3.1.3.2. 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.

3.1.3.3. Criterio de aceptación

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

3.1.3.4. 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.

3.1.4. Prueba de Interfaz de Usuario

Esta prueba verifica que la interfaz de usuario proporcione al usuario el acceso y navegación a través de las funciones apropiada. Además asegura que los objetos presentes en la interfaz de usuario se muestren como se

espera y conforme a los estándares establecidos por la empresa o de la industria.

3.1.4.1. 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.

3.1.4.2. 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.

3.1.4.3. Criterio de aceptación

Cada ventana ha sido verificada exitosamente siendo consistente con una versión de referencia o estándar establecido.

3.1.4.4. Consideraciones especiales

No todas las propiedades de los objetos se pueden acceder.

3.1.5. Prueba de Performance

En esta prueba se miden y evalúan los tiempos de respuesta, los tiempos de transacción y otros requerimientos sensitivos al tiempo. El objetivo de la prueba es verificar que se logren los requisitos de performance. La prueba de performance es implementada y ejecutada para poner a punto los destinos de pruebas de performance como función de condiciones de trabajo o configuraciones de hardware. En cuanto a este tipo de pruebas el cliente solicitó que el sistema tuviera tiempos de respuesta razonables. Antes de realizar este tipo de pruebas debemos definir métricas para medir lo solicitado por el cliente y evaluar la posibilidad de realizar las pruebas teniendo en cuenta que el ambiente de testing puede variar respecto al ambiente de producción. Por ello se evaluará en las reuniones de la próxima semana con el cliente.

3.1.5.1. Objetivo de la prueba

Verificar la performance de determinadas transacciones o funciones de negocio bajo ciertas condiciones:

- condiciones de trabajo normales conocidas.
- peores casos de condiciones de trabajo conocidas.

3.1.5.2. Técnica

- Usar procedimientos de prueba desarrollados para verificar funciones o ciclos de negocio.
- Modificar archivos de datos para aumentar el número de transacciones o los procedimientos de prueba para aumentar el número de iteraciones de ocurrencia de transacciones.
- Las pruebas se deben ejecutar en una máquina (mejor caso de prueba un solo usuario, una sola transacción) y se debe repetir con múltiples usuarios (virtuales o reales).

3.1.5.3. Criterio de aceptación

Con una transacción o un usuario: Éxito completo de la prueba sin fallas y dentro del tiempo esperado o requerido.

Con múltiples transacciones y varios usuarios: Éxito completo de la prueba sin fallas y dentro de un tiempo aceptable.

3.1.5.4. Consideraciones especiales

Las pruebas de performance deben incluir un trabajo de fondo en el servidor. Esto se puede realizar de distintas formas:

- Enviar transacciones directamente al servidor, generalmente en la forma de consultas (SQL).
- Crear usuarios virtuales para simular muchos clientes, generalmente varios cientos. Se pueden usar herramientas de Emulación de Terminal Remota para lograr este objetivo. Esta técnica también se usa para cargar la red con "tráfico".
- Usar muchos clientes físicos, cada uno corriendo procedimientos de prueba.

La prueba de performance se debe realizar en una máquina dedicada para permitir control total y medición exacta.

Las bases de datos usadas para las pruebas de performance deben tener un tamaño similar a las reales.

3.1.6. Prueba de Carga

La prueba de carga somete los objetos a verificar a diferentes cargas de trabajo para medir y evaluar los comportamientos de performance y la habilidad de los objetos de continuar funcionando apropiadamente bajo diferentes cargas de trabajo. El objetivo es determinar y asegurar que el sistema funciona apropiadamente en circunstancias de máxima carga de trabajo esperada. Además evaluar las características de performance, como tiempos de respuesta, tiempos de transacciones y otros elementos sensitivos al tiempo. Igual que en el caso anterior se deberán definir métricas para medir tiempos "razonables".

3.1.6.1. Objetivo de la prueba

Verificar el comportamiento de performance de determinados componentes del software bajo condiciones de trabajo diferentes.

3.1.6.2. Técnica

Usar pruebas desarrolladas para funciones o ciclos de negocios y modificar archivos de datos para aumentar el número de transacciones o las pruebas para aumentar la cantidad de ocurrencia de transacciones.

3.1.6.3. Criterio de aceptación

Para múltiples transacciones y múltiples usuarios: Realización exitosa de las pruebas sin fallas y dentro del tiempo aceptable.

3.1.6.4. Consideraciones especiales

La prueba de carga debe realizarse en una máquina dedicada para tener control total y exactitud de mediciones.

Las bases de datos usadas para la prueba deben tener un tamaño similar a las reales.

Dado que el cliente no prevé una gran carga sobre el sistema, este tipo de testing no será realizado.

3.1.7. Prueba de Esfuerzo (stress, competencia por recursos, bajos recursos)

La prueba de esfuerzo es un tipo de prueba de performance implementada y ejecutada para encontrar errores cuando hay pocos recursos o cuando hay competencia por recursos. Poca memoria o poco espacio de disco pueden revelar fallas en el software que no aparecen bajo condiciones normales de cantidad de recursos. Otras fallas pueden resultar al competir por recursos compartidos como bloqueos de bases de datos o ancho de banda de red. La prueba de esfuerzo también puede usarse para identificar el trabajo máximo que el software puede manejar.

3.1.7.1. Objetivo de la prueba

Verificar que el software funciona apropiadamente y sin error bajo condiciones de esfuerzo, como son:

- poca memoria o sin disponibilidad de memoria en el servidor
- cantidad máxima de clientes conectados
- múltiples usuarios realizando la misma operación sobre los mismos datos
- peor caso de volumen de operaciones.

El objetivo de la prueba de esfuerzo es también identificar y documentar las condiciones bajo las cuales el sistema falla y no continua funcionando apropiadamente.

3.1.7.2. Técnica

Usar las pruebas desarrolladas para Performance y Prueba de Carga.

Para probar recursos limitados, las pruebas se deben ejecutar en una sola máquina, y se debe reducir o limitar la memoria en el servidor.

Para las pruebas de esfuerzo restantes, deber usarse múltiples clientes, cualquiera que ejecute las mismas pruebas o pruebas complementarias para producir el peor caso de volumen de operaciones.

3.1.7.3. Criterio de aceptación

Todas las pruebas planeadas se ejecutaron y se alcanzaron o excedieron los límites del sistema sin que el software fallara o las condiciones bajo las que ocurre una falla en el software están fuera de las condiciones especificadas.

3.1.7.4. Consideraciones especiales

Las pruebas de esfuerzo de red pueden requerir herramientas de red para cargar la red con mensajes o paquetes.

La cantidad de disco del servidor usada por el sistema debe ser reducida temporalmente para restringir el espacio disponible para crecimiento de la base de datos.

Sincronizar el acceso simultáneo de varios clientes accediendo a los mismos datos.

Dado que el cliente no prevé una gran carga sobre el sistema, este tipo de testing no será realizado.

3.1.8. Prueba de Volumen

La Prueba de Volumen somete el software a grandes cantidades de datos para determinar si se alcanzan límites que causen la falla del software. La Prueba de Volumen identifica la carga máxima continua que puede manejar el software a prueba en un período dado.

3.1.8.1. *Objetivo de la prueba*

Verificar que el software funciona correctamente con volúmenes de datos grandes:

- Máximo (real o físicamente posible) número de clientes conectados, o simulados, todos realizando la misma operación (peor caso de operación) por un período de tiempo extenso.
- Máximo tamaño de base de datos y múltiples consultas ejecutadas simultáneamente.

3.1.8.2. *Técnica*

Usar pruebas desarrolladas para Prueba de Performance y Prueba de Carga.

- Se deben usar múltiples clientes, ejecutando las mismas pruebas o pruebas complementarias para producir el peor caso de volumen de operaciones o mezcla en un período de tiempo extenso.
- Se debe crear el tamaño máximo de base de datos (real, escalado o con datos representativos) y múltiples clientes ejecutando consultas simultáneamente por un período de tiempo extenso.]

3.1.8.3. *Criterio de aceptación*

Todas las pruebas planificadas se ejecutaron y se han alcanzado o excedido los límites especificados sin que el software falle.

3.1.8.4. *Consideraciones especiales*

¿Qué período de tiempo se considera aceptable para condiciones de gran volumen?

Dado que el cliente no prevé un volumen importante de datos sobre el sistema, este tipo de testing no será realizado.

3.1.9. 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.

3.1.9.1. *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.

3.1.9.2. *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.

Acceso en el ámbito de sistema: Ver consideraciones especiales más abajo.

3.1.9.3. *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.

3.1.9.4. *Consideraciones especiales*

El acceso al sistema debe ser discutido con el administrador del sistema o la red. Esta prueba no puede requerirse como tal, es una función del administrador del sistema o de la red.

3.1.10. Prueba de Fallas y Recuperación

Las Pruebas de Fallas y Recuperación aseguran que el software puede recuperarse de fallas de hardware, software o mal funcionamiento de la red sin pérdida de datos o de integridad de los datos.

La Prueba de Recuperación es un proceso en el cual la aplicación o sistema se expone a condiciones extremas, o condiciones simuladas, para causar falla, como fallas en dispositivos de Entrada/Salida o punteros a la base de datos inválidos. Los procedimientos de recuperación se invocan y la aplicación o sistema es monitoreado e inspeccionado para verificar que se recupera apropiadamente la aplicación o sistema y se logre la recuperación de datos.

3.1.10.1. *Objetivo de la prueba*

Verificar que los procesos de recuperación (manual o automáticos) recuperen apropiadamente la base de datos, aplicaciones y sistema a un estado conocido y deseado. En la prueba se incluyen los siguientes tipos de condiciones:

- interrupción de energía al cliente
- interrupción de energía al servidor
- interrupción de comunicaciones mediante los servidores de la red
- interrupción de comunicación o pérdida de energía de los discos del servidor o con los controladores
- ciclos incompletos (procesos de filtro de datos interrumpidos, procesos de sincronización de datos interrumpidos)
- punteros a la base de datos o claves inválidos
- elementos de datos en la base de datos inválidos o corruptos.]

3.1.10.2. *Técnica*

Se deben usar las pruebas creadas para probar Funcionalidad y Ciclos de negocio para crear una serie de operaciones. Una vez logrado el punto de comienzo deseado, se deben realizar o simular las siguientes acciones, individualmente:

- Interrumpir la energía del cliente: apagar el PC.
- Interrumpir la energía del servidor: simular o iniciar el proceso de apagado del servidor.
- Interrupción por medio de los servidores de red: simular o iniciar la pérdida de comunicación con la red (desconectar físicamente la comunicación o apagar el servidor de red o router
- Interrumpir la comunicación o quitar la energía de los discos del servidor o sus controladores: simular o eliminar físicamente al comunicación con uno o más controladores de disco o los discos.
- Una vez que se lograron o simularon estas condiciones, se deben invocar los procedimientos de recuperación.
- Las pruebas de ciclos incompletos utilizan la misma técnica excepto que los procesos de bases de datos deben ser abortados a sí mismos o terminados prematuramente.
- Las últimas dos pruebas requieren que se logre un estado conocido de la base de datos. Se deben corromper manualmente campos de la base de datos, punteros y claves trabajando directamente sobre la base de datos (utilizando herramientas para la base de datos). Se deben ejecutar las pruebas de Funcionalidad y Ciclo de negocio y verificar que los ciclos se completen.

3.1.10.3. Criterio de aceptación

En todos los casos, la aplicación, la base de datos y el sistema deben, en la realización procedimientos de recuperación, volver a un estado conocido y deseable. Este estado incluye corrupción de datos limitada al los campos, punteros o claves corruptos conocidos, y reportes indicando los procesos u operaciones que no se completaron debido a las interrupciones.

3.1.10.4. Consideraciones especiales

Los procedimientos para desconectar cables (simulando falta de energía o pérdida de comunicación) no son deseables o factibles. Se pueden requerir métodos alternativos, como software de diagnóstico. Se requieren los grupos de recursos de Sistemas, Bases de datos y Red.

Estas pruebas deben ejecutarse fuera del horario de trabajo normal o en una máquina aislada.

3.1.11. Prueba de Configuración

La Prueba de Configuración verifica el funcionamiento del software con diferentes configuraciones de software y hardware.

3.1.11.1. Objetivo de la prueba

Verificar que el software funcione apropiadamente en las configuraciones requeridas de hardware y software.

3.1.11.2. Técnica

Usar las pruebas de Funcionalidad.

- Abrir y cerrar varias sesiones de software que no son objeto de prueba, como parte de la prueba o antes de comenzar la prueba.

- Ejecutar operaciones seleccionadas para simular la interacción del actor con el software objeto de prueba y con el software que no es objeto de prueba.
- Repetir los procedimientos anteriores minimizando la memoria convencional disponible en la máquina cliente.

3.1.11.3. *Criterio de aceptación*

Por cada combinación de software objeto de prueba y software que no es objeto de prueba, todas las operaciones son completadas exitosamente sin fallas.

3.1.11.4. *Consideraciones especiales*

Todo el software que no es objeto de prueba que es necesario y debe estar accesible.

¿Qué aplicaciones se usan normalmente?

¿Qué información se maneja en las aplicaciones que se usan normalmente, y que tamaño de información?

Los sistemas, red, servidores de red, bases de datos, etc., deben ser documentados como parte de esta prueba.

3.1.12. Prueba de Instalación

La Prueba de Instalación tiene dos propósitos. Uno es asegurar que el software puede ser instalado en diferentes condiciones (como una nueva instalación, una actualización, y una instalación completa o personalizada) bajo condiciones normales y anormales. Condiciones anormales pueden ser insuficiente espacio en disco, falta de privilegios para crear directorios, etc. El otro propósito es verificar que, una vez instalado, el software opera correctamente. Esto significa normalmente ejecutar un conjunto de pruebas que fueron desarrolladas para Prueba de Funcionalidad.

3.1.12.1. *Objetivo de la prueba*

Verificar que el software objeto de prueba se instala correctamente en cada configuración de hardware requerida bajo las siguientes condiciones:

- instalación nueva, una nueva máquina, nunca instalada previamente con GVA.
- actualización, máquina previamente instalada con GVA, con la misma versión
- actualización, máquina previamente instalada con GVA con una versión anterior.

3.1.12.2. *Técnica*

Manualmente o desarrollando programas, para validar la condición de la máquina destino (nueva, nunca instalado, misma versión, versión anterior ya instalada).

Realizar la instalación.

Ejecutar un conjunto de pruebas funcionales ya implementadas para la Prueba de Funcionalidad.

3.1.12.3. *Criterio de aceptación*

Las pruebas de funcionalidad de GVA se ejecutan exitosamente sin fallas.

3.1.12.4. *Consideraciones especiales*

¿Qué operaciones se deben seleccionar para realizar una prueba confiable de que la aplicación GVA ha sido exitosamente instalada sin dejar fuera ningún componente importante?

3.1.13. 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, Ayuda en Línea y todo tipo de documento que forme parte del paquete de software.

3.1.13.1. Objetivo de la prueba

Verificar que el documento objeto de prueba sea:

- Correcto, esto es, que cumpla con el formato y organización para el documento 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.

3.1.13.2. Técnica

Para verificar que el documento es correcto se debe comparar con el estándar definido si existe 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.

3.1.13.3. 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.

3.1.13.4. Consideraciones especiales

Enumere las consideraciones que considere importantes para la verificación de documentos.

3.2. Herramientas

A continuación se presenta una lista con las herramientas usadas en el proyecto:

MySQL: se utilizará para la gestión de sistema de bases de datos.

Google Groups: esta herramienta se utilizará para la gestión del proyecto.

Google Drive: dicha herramienta se empleará para la gestión de los documentos, carga de horas del equipo, etc.

Mantis BugTracker: herramienta para el reporte y seguimiento de incidencias.

SVN: dicha herramienta se utilizará para el control de versiones del código fuente.

4. Recursos

En esta sección se presentan los recursos recomendados para el proyecto GVA 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 GVA en el área Verificación del Software.

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

4.2. Sistema

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

Es recomendable que el sistema simule el entorno de producción, reduciendo los accesos y los tamaños de bases de datos si fuera apropiado.

| Recurso | Nombre/Tipo |
|--------------------------------------|--------------------|
| Servidor de base de datos | MySQL |
| Herramienta para seguimiento de bugs | Mantis |
| Red o subred | |
| Nombre del servidor | |
| Nombre de la base de datos | |
| PC Cliente para pruebas | |
| Requerimientos especiales | |
| Repositorio de pruebas | |
| Red o subred | |
| Nombre del servidor | |

5. Hitos del proyecto de Verificación

La verificación de GVA 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.

| Actividad que determina el hito | Esfuerzo | Fecha de comienzo | Fecha de finalización |
|--|-----------------|--------------------------|------------------------------|
| Planificar la verificación | 15 | Semana 2 | Semana 4 |
| Elaborar casos de prueba | 20 | Semana 3 | Semana 12 |
| Ajuste y Control de Verificación | 10 | Semana 9 | Semana 11 |
| Ejecutar la verificación | 30 | Semana 4 | Semana 13 |
| Evaluar la verificación | 15 | Semana 5 | Semana 13 |

6. Entregables

6.1. Modelo de Casos de Prueba

| | |
|---------------------|--|
| Documento | Modelo de Casos de Prueba |
| Creado por | Christopher Quincke |
| Para quien | Es la guía para realizar las pruebas del sistema y lo usarán los Asistentes de verificación y el Responsable de verificación cuando se ejecuten las pruebas del sistema. |
| Fecha de liberación | Será liberado al fin de la semana 4. |

6.2. Informes de Verificación

| | |
|------------|--|
| Documento | Se genera un documento Informe de Verificación Unitaria 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. Elaboración, 1era iteración: Semana 06 Elaboración, 2da iteración: Semana 08 Construcción, 1era iteración: Semana 10 Construcción, 2da iteración: Semana 12 Transición, 1era iteración: Semana 14 |

| | |
|---------------------|---|
| Documento | Se genera un documento Informe de Verificación de Integración 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. Elaboración, 1era iteración: Semana 06 Elaboración, 2da iteración: Semana 08 Construcción, 1era iteración: Semana 10 Construcción, 2da iteración: Semana 12 Transición, 1era iteración: Semana 14 |

6.3. Evaluación de la verificación

| | |
|---------------------|---|
| Documento | Se genera un documento Evaluación de la verificación 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. Elaboración, 1era iteración: Semana 06 Elaboración, 2da iteración: Semana 08 Construcción, 1era iteración: Semana 10 Construcción, 2da iteración: Semana 12 Transición, 1era iteración: Semana 13 |

6.4. Informe final de verificación

| | |
|-----------|---|
| Documento | El documento Informe final de verificación 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

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

El equipo de verificación está integrado por el encargado de verificación y cuatro asistentes de verificación. Los asistentes de verificación tienen otros roles asignados de mayor prioridad pero de todas formas deberán contribuir a la verificación del sistema.

8. Riesgos

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

8.1. Planificación

Dado que la mayoría de los integrantes del equipo de verificación tiene otros roles mas prioritarios un riesgo es que no puedan dedicar el tiempo planificado a la 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 los 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.