WIRA

Plan de Verificación y Validación

Versión 1.2.1

Semana 06

Grupo 11

Historia de revisiones

Fecha	Versión	Descripción	Autor
31/08/2013	1.0	Creación del documento	Juan Bertoni
01/09/2013	1.0.1	Revisión SQA	Gonzalo Antúnez
21/09/2013	1.1	Se agregan pruebas a realizar en el alcance	Juan Bertoni
22/09/2013	1.1.1	Revisión SQA	Gonzalo Antúnez
29/09/2013	1.2	Se agregaron requerimientos no funcionales	Juan Bertoni
29/09/2013	1.2.1	Revisión SQA	Gonzalo Antúnez

Contenido

1. Introducción

1.1. Propósito

Este Plan de Verificación para el proyecto 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 considerando las prioridades que el cliente establezca en cuanto a la importancia de cada requerimiento y el impacto del mismo dentro de la aplicación
- Identificar los recursos necesarios y proporcionar una estimación de esfuerzo para realizar la verificación.
- Enumerar los entregables del proyecto de verificación.
- Recomendar y describir las estrategias de verificación que serán usadas

1.2. Punto de partida

El objetivo de la verificación de software es identificar y asegurar que sean corregidos los defectos antes de la liberación de la versión del producto. Se debe comprobar que se satisfagan los requerimientos funcionales y no funcionales, así como también garantizar que la integración de los componentes que forman el sistema están correctamente integrados.

El Proyecto se puede dividir en 2 objetivos, uno es el de brindar una forma sencilla de almacenar los datos de la estancia y generar reportes para los organismos del estado. El otro es brindar información del uso de los usuarios, los cuales en su mayoría no están acostumbrados al uso de software en general. Para facilitar esta ultima funcionalidad se utilizará una herramienta de de medición como Google Analytics. La aplicación correrá en en las ultimas versiones de Firefox, Chrome e Intenet Explorer de escritorio.

Al tener una arquitectura MVC se verificará el correcto funcionamiento de cada capa (y de sus respectivas partes) tanto unitariamente como integradas, para ello se implementarán pruebas acordes a éste tipo de arquitectura y a cada capa. La validación tendrá lugar sobre el sistema en general aunque se planeen pruebas de aceptación de ciertas funcionalidades importantes para el cliente.

La verificación :

- se realiza durante todo el proceso de desarrollo de software e incluye a todos los elementos que componen el sistema, como son, materiales para soporte al usuario, documentación técnica, software y sistema.
- de los documentos se hace a medida que éstos se van generando y al final de la construcción de los mismos.
- del software se realiza en el entorno de desarrollo cada vez que se integra algún componente al sistema.
- se realiza cuando se genera un ejecutable del sistema y se verifica éste en el entorno del usuario.

1.3. Alcance

Se definen las siguientes fases para el proceso de verificación, siguiendo el orden en el cual están presentados:

- **Unitaria:** se verifican los componentes del sistema en forma unitaria. Esta verificación será realizada por los implementadores. Estos deberán verificar cada componente que construyen y elaborar los informes de verificación unitaria correspondientes. Estos últimos deben contener la cantidad de errores encontrados y los que fueron

corregidos en cada iteración. La importancia de la verificación a este nivel es mayor, dado que la introducción de defectos en esta etapa puede afectar las pruebas de verificación siguientes.

- **Integración:** con los componentes previamente verificados, se prueba la interacción entre estos teniendo siempre en cuenta los requerimientos a cumplir.
- **Pruebas del sistema:**
  - Funcional: se verifica que el sistema cumpla con las funcionalidades descritas en los requerimientos del cliente, los cuales están dentro del alcance del producto.
  - Ciclo de negocio: se hará la prueba de que los eventos hagan envíos automáticos de mails y actualicen los datos cuando corresponda
  - Performance: se verifica que los tiempos de navegación estén por debajo de los 5 segundos.
  - Interfaz: se verifica que se respeten las navegabilidades entre los casos de uso establecido en el modelo de casos de uso. También se verifica que se cumpla las pautas de interfaz acordadas con el cliente
- Aceptación: se verifica que el sistema cumple con lo que el cliente deseaba o pretendía. Es realizada por el cliente mismo o en su defecto el grupo de usuarios finales del producto.

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

- Agregar al producto requerimientos no pactados en el alcance del proyecto. Esto dependerá de una buena validación de los requerimientos 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 (Ruby on Rails, Google Analytics, Mantis Bug Tracker, etc.).

1.4. Identificación del proyecto

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

- Template del Plan de Verificación y Validación especificada en el Modelo de Proceso Modularizado Unificado y Medible (MUM) utilizado durante este proyecto.
- 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.
- Documento de Especificación de Requerimiento.

1.5. Estrategia de evolución del Plan

El responsable de monitorear el Plan de Verificación y Validación es el responsable de verificación. Durante las dos primeras fases deberá planificar la verificación que se hará a lo largo del proyecto.

Los cambios del Plan serán evaluados y aprobados por el responsable de verificación en conjunto con todos los asistentes, pudiendo estos cambios ser sugeridos por todos los integrantes del equipo.

Es deber del responsable de Verificación informar al resto del grupo los ajustes realizados al plan, mediante las herramientas de comunicación definidas para el proyecto.

2. Requerimientos para verificar

En la lista a continuación se presentan los elementos, casos de uso, requerimientos funcionales y requerimientos no funcionales, que serán verificados.

Para la versión final del producto se verificarán todos los requerimientos funcionales y no funcionales presentados en el documento de especificación de requerimientos**,** que hayan caído dentro del alcance final pactado con el cliente. En el transcurso del proceso se priorizarán requerimientos para verificar, que dependerán de la iteración y fase en la que se encuentra el producto.
En la lista a continuación se presentan los elementos, casos de uso, requerimientos funcionales y requerimientos no funcionales, que serán verificados.

No Funcionales:

- seguridad de la web.
- concurrencia de usuarios
- estética de las interfaces gráficas.
- tiempos de espera aceptables de la web
- versión web: firefox 23+, IE 10+ o chrome 29+, navegador de XO
- Tamaño minimo de pantalla para uso de 640×350 con botones no menores a 45 pixeles

3. Estrategia de Verificación

Esta sección describe los tipos de pruebas que se realizaran con sus objetivos, técnicas y criterios de aceptación, y las herramientas necesarias.

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.

3.1.1.2. Técnica

Probar con distintos tipos de datos, hacer pruebas con datos válidos e inválidos. Revisar la base de datos para comprobar que los datos son consistentes, y que el comportamiento es el esperado.

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.

3.1.2. Prueba de Funcionalidad

La prueba de funcionalidad se enfoca en requerimientos para verificar que se corresponden directamente a casos de usos ó 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

Ejecución de 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

No hay

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 menúes, 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 requerimientos 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.

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 Terminar 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.

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 ó 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.

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

La prueba de esfuerzo en 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.

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?

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:
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.

3.1.9.3. 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 ó claves inválidos
- elementos de datos en la base de datos inválidos ó 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

Al tratarse de una aplicación web, se tiene que verificar la instalación en el ambiente de desarrollo.

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 entendibles. 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

No hay

3.2. Herramientas

Se esta evaluando el uso de Mantis Bug Tracker versión 1.2.15 para llevar el registro de los defectos encontrados, manteniendo organizadas las acciones a tomar sobre estos.

También se esta evaluando a la herramienta Simplecov para el monitoreo de cubrimiento de código.

4. Recursos

En esta sección se presentan los recursos recomendados para el proyecto, 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 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. • 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	3	• 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	• 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.

Recurso	Nombre/Tipo
Servidor de base de datos	PostgreSQL
Red o subred	Internet
Nombre del servidor	
Nombre de la base de datos	
PC Cliente para pruebas	
Requerimientos especiales	
Repositorio de pruebas	Repositorio Git
Red o subred	
Nombre del servidor	

5. Hitos del proyecto de Verificación

La verificación del proyecto 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

En esta sección se enumeran los documentos, herramientas e informes que se crearán, por quien, para quien y cuándo serán liberados.
Para cada entregable se indica las fechas en que son liberadas todas las versiones del mismo.

6.1. Modelo de Casos de Prueba

Documento	**Modelo de Casos de Prueba**
Creado por	El Responsable de verificación, Juan Bertoni.
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.

Documento	Se genera un documento **Informe Consolidación** 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 **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.

Documento	Se genera un documento **Informe de Verificación de Sistema** 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 **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.

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. Apéndice

7.1. 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.

7.2. Niveles de aceptación para lo elementos verificados

En esta sección se define los niveles de aceptación y los criterios de pertenencia a cada nivel:

- **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.