Proyecto Moove-IT

Plan de Verificación y Validación

Versión [4.0]

Historia de revisiones

Fecha	Versión	Descripción	Autor
02/09/13	1.0	Fase Inicial – Primera iteración	Nicolás Ramponi
02/09/13	1.1	Revisión de SQA	Ezequiel Jardim
15/09/13	4.0	Fase Inicial – Segunda iteración	Nicolás Ramponi
15/09/13	4.1	Revisión de SQA	Ezequiel Jardim

Contenido

1. Introducción

1.1. Propósito

Este Plan de Verificación para el proyecto Moove-IT 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

El objetivo de la implementación de nuestro plan de verificación radica en permitir brindar confianza y seguridad en la funcionalidad y estabilidad del producto de software que se intenta producir en este proyecto.

El software debería poder brindar al cliente una herramienta de acceso rápido para ingresar horas trabajadas y estimaciones en distintos servicios de manejo de proyectos de desarrollo de software. Para ello, el proyecto Moove-IT debe proveer tres interfaces: un servidor web, un plugin de Google Chrome y una aplicación móvil. Todas estas interfaces deben proveer un manejo veloz (no más de 1 segundo de respuesta en un ambiente de hasta 100 usuarios simultáneos) por lo tanto se requiere de un Plan de Verificación que asegure performance, además de la adecuación a distintos ambientes (por ejemplo, la aplicación móvil debe correr en iOS y Android).

1.3. Alcance

Dado que en años anteriores la realización de pruebas previo a la realización del código (Test First Development) funciono de manera caótica, se decidió encontrar una nueva manera de proceder a hacer el testing. La misma se centrará en la capacidad de los desarrolladores de encontrar errores en el código.

Luego de discutir el tema con los implementadores del equipo, se optó por un método de verificación en parejas que divide las tareas de implementación y verificación unitaria entre grupos de dos implementadores. Entonces, cada integrante de la pareja hace dos cosas: genera su propio código y testea el código de su compañero.

Esto permitiría que el código que genera un compañero sea verificado por el otro y viceversa, logrando las siguientes ventajas:

- Más Disciplina. Al no poder testear el código hasta que el mismo no esté completado por su compañero, el tester de la dupla se ocupa de presionar al implementador para que termine su tarea a tiempo.

- Mejor código. La necesidad de generar código que sea legible por una persona externa que busca defectos fomenta el buen diseño.

También ayuda a que el ego del implementador no bloquee su capacidad de ver errores propios en el código, ya que va a ser una persona externa la que se ocupe de revisar el mismo.

- Mejor conocimiento del código. Cuando la implementación se hace en parejas, (más si las parejas se rotan con frecuencia), todos terminan teniendo un conocimiento del código base.

- Enseñanza. Ya que van a haber dos personas mirando el mismo código, pueden existir instancias de transmisión de conocimiento, ayudando a la enseñanza de habilidad de programación, fácilmente generando programadores con mejores capacidades.

- Cohesión de equipo. La gente se familiariza más rápidamente cuando verifica en parejas. La verificación en parejas puede animar el sentimiento de equipo ya que se generan mas instancias de comunicación personal.

Se constató también que la programación en pareja puede ser mejorada si se agrega la rotación de parejas. Se plantea que al principio de cada semana, se le asigne a cada integrante del grupo su pareja para el resto de la misma.

La rotación de parejas puede aportar las siguientes ventajas:

- Si una pareja se mantiene a lo largo de todo el periodo de implementación, el tester se puede acostumbrar a la manera de codificar de su pareja implementadora, lo que termina deteriorando el diseño que debe permitir buena visualización para todo el que quiera consultar.

- Una nueva persona todas las semanas puede encontrar fallas que la persona anterior no había visto.

- Permite que todo implementador tenga contacto individual al menos una vez en el proceso con todos los otros implementadores, permitiendo una mejor comunicación y sentimiento de equipo.

- Las enseñanzas fluirían de una forma más homogénea en todo el grupo.

Lamentablemente, luego de discutir en profundidad sobre el tema con el equipo de implementadores, se decidió que no iba a ser muy productivo llevar a cabo esta rotación, ya que el proyecto existe la necesidad de actuar rápido con la implementación del software. Cambiar de pareja todas las semanas enlentecería el proceso, que puede funcionar con menos trabas si se mantiene el mismo equipo durante todo el proceso.

1.4. Identificación del proyecto

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

- Teórico del curso Taller de Verificación de Software
- La "Guía respecto a la Programación por Pares" del Departamento de Informática de la Universidad Técnica Federico Santa María de Chile
- Plan de verificación y validación de otros años tomado de la Memoria Organizacional de la asignatura.

- Plantilla del Plan de Verificación y Validación proporcionada en el Modelo de Proceso Modularizado Unificado y Medible (MUM) utilizado durante este proyecto.

1.5. Estrategia de evolución del Plan

- El responsable de monitorear el Plan de Verificación y Validación es el responsable de Verificación, con el apoyo de los asistentes de Verificación.
- El plan será revisado y modificado si es necesario, una vez cada dos semanas, o sea, una vez cada iteración.
- Cualquier cambio que se haga al plan será evaluado en conjunto con los asistentes de verificación y luego con el equipo de implementadores. Desde un punto de vista de calidad, el documento será verificado por el responsable de SQA.
- En caso de haber necesidad de cambiar cualquier aspecto del plan, estos serán comunicados al grupo de implementadores en reuniones planificadas.

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.

Casos de uso:

- Los casos de uso varían a medida que avanza el proyecto (al menos esto es lo que se considera como verdadero en el desarrollo agil). Por lo tanto, se va a preveer una planificación de la verificación antes de cada Sprint de los casos de uso (historias) que se van a implementar.

Requerimientos Funcionales:

- El software debe permitir a todos los usuarios ingresar datos correspondientes a las horas que planifica y realiza en cualquier proyecto del que forme parte de forma intuitiva y rápida.
- Éste debe interactuar con las interfaces de las distintas herramientas de planificación de proyectos (Jira, Pivotal Tracker, Assembla, Feng Office y Redmine)
- El software debe permitir acceder a un Backend donde se encuentre toda la información para que esta pueda ser visualizada en forma de gráficas y reportes

Requerimientos No Funcionales:

- El software debe estar verificado con un cubrimiento mayor al 90% del código, asegurando un cierto nivel de calidad
- El tiempo de respuesta para cualquiera de las interfaces (Móvil, Servidor, Plugin de Chrome) debe ser menor de 1 segundo, incluso en ambientes de hasta 100 usuarios simultáneos.

3. Estrategia de Verificación

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

Se invocará cada proceso de acceso a la base de datos con datos válidos y no válidos para testear su funcionalidad y reacción ante errores.

Luego se procederá a inspeccionar la base de datos para asegurar que se han guardado los datos correctos, que todos los eventos de la base de datos ocurrieron correctamente, o se repasan 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, permitiendo también que las pruebas no tomen mucho tiempo.

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

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.

Estas pruebas de funcionalidad se realizaran por parte del equipo de verificación compuesto por el Responsable de Verificación y asistentes. Además del Testing Planificado basado en casos de prueba, se procederá también en lo que se denomina como Testing Exploratorio basado en sesiones. Para este tipo de test se dedicará una hora a la semana para cada miembro, que ejecutara el programa con las funcionalidades que provea hasta ese momento, lo utilizará de manera desorganizada e intuitiva, y documentara las sesiones correspondientemente.

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.

Las pruebas funcionales tienen que pasar en un 100% de los casos, y todas las fallas encontradas deben ser mitigadas. También, el código debe cubrirse en un 90%.

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 tiempos 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 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 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 "trafico".
- 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 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.

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: 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 Proyecto Moove-IT.
- actualización, máquina previamente instalada con Proyecto Moove-IT con la misma versión.
- actualización, máquina previamente instalada con Proyecto Moove-IT 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 del Proyecto Moove-IT se ejecutan exitosamente sin fallas.

3.1.12.4. Consideraciones especiales

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

3.2. Herramientas

- Selenium Web Driver y Selenium Grid : Realización de pruebas automatizadas para una aplicación web usando scripts (aún queda verificar si puede hacer test concurrentes)
- Jmeter : Pruebas de performance para grandes cantidades de solicitudes a un servidor web que permite verificar si el tiempo de respuesta de una solicitud está dentro de lo aceptable
- Simplecov : se utiliza para evaluar el cubrimiento de código de los tests realizados

Se utilizará un documento Excel para monitorear las sesiones y misiones del testing exploratorio, que debe poder realizarse en los ambientes establecidos por el cliente del proyecto.

También se utilizara un Excel en Google Drive para reportar todas las pruebas e indicar errores encontrados, ayudando a la comunicación y efectividad en la verificación. Se separarán las pruebas de los distintos tipos en hojas de cálculo, para permitir dividir responsabilidades y priorizar los distintos tipos de pruebas acorde con este proyecto.

4. Recursos

En esta sección se presentan los recursos recomendados para el proyecto Moove-IT 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 Moove-IT 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.

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

[Borre o agregue elementos a la lista]

Recurso	Nombre/Tipo
Servidor de base de datos	MySQL
Red o subred	Falta definir
Nombre del servidor	Falta definir
Nombre de la base de datos	Falta definir
PC Cliente para pruebas	Falta definir
Requerimientos especiales	Falta definir
Repositorio de pruebas	Falta definir
Red o subred	Falta definir
Nombre del servidor	Falta definir

5. Hitos del proyecto de Verificación

Actividad que determina el hito	Esfuerzo Hs/semana	Fecha de comienzo	Fecha de finalización
Iteración 1 – Fase Inicial – 1 y 2			
Planificar la verificación	14	26/08/2013	01/09/2013
Iteración 2 – Fase Inicial – 3 y 4			
Elaborar casos de Prueba	5	02/09/2013	15/09/2013
Ajuste y Control de Verificación	5	02/09/2013	15/09/2013
Ejecutar la verificación (prototipo)	7	09/09/2013	15/09/2013
Evaluar la verificación (prototipo)	7	09/09/2013	15/09/2013
Iteración 1 – Elaboración – 5 y 6			
Ajuste y Control de Verificación	10	16/09/2013	29/09/2013
Ejecutar la verificación	24	16/09/2013	29/09/2013
Evaluar la verificación	20	16/09/2013	29/09/2013
Iteración 2 – Elaboración – 7 y 8			
Ajuste y Control de Verificación			
Ejecutar la verificación			
Evaluar la verificación			

Iteración 1 – Construcción – 9 y 10			
Ajuste y Control de Verificación			
Ejecutar la verificación			
Evaluar la verificación			
Iteración 2 – Construcción – 11 y 12			
Ajuste y Control de Verificación			
Ejecutar la verificación			
Evaluar la verificación			
Iteración 1 – Transición - 13			
Ajuste y Control de Verificación			
Ejecutar la verificación			
Evaluar la verificación			
Iteración 2 – Transición - 14			
Ajuste y Control de Verificación			
Ejecutar la verificación			
Evaluar la verificación			

6. Entregables

6.1. Modelo de Casos de Prueba

Documento	**Modelo de Casos de Prueba**
Creado por	El Responsable de verificación, Nicolás Ramponi
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 el 18/09/2013

6.2. Informes de Verificación

Documento	Se genera un documento **Informe de Verificación de Integración** por cada prueba de integración que se realice al sistema.
Creado por	La persona del equipo de verificación que realice 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. La fecha de realización de la integración aún no está definida.

Documento	Se genera un documento **Informe de Verificación de Sistema** por cada prueba de sistema que se realice.
Creado por	La persona del equipo de verificación que realice 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. La fecha de liberación aún no está definida.

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. La fecha de liberación aún no está definida.

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

Para la realización y ejecución de la verificación, se va necesitar, además del responsable de verificación, un equipo de 2 o 3 asistentes. Debido a la cantidad de otras responsabilidades que estas personas tienen por dedicación a otros roles en el proyecto, se debe planificar una semana antes cual va a ser la cantidad de tiempo disponible que tengan estos asistentes para colaborar.

7.2. Dependencia de software

Se debe tener un ambiente de realización de pruebas con todas las herramientas propiamente instaladas y probadas para así generar experiencia en su uso, evitando problemas en el futuro cuando la implementación comience.

El ambiente debe ser el mismo que va a utilizar el cliente (mismo sistema operativo por ejemplo). Este normalmente va a diferir del ambiente donde se realicen las pruebas unitarias por parte de los implementadores.

7.3. Dependencia de hardware

Se debe disponer de instrumental necesario para realizar las pruebas en el ambiente que el cliente requiere. En este proyecto, se requiere conseguir un dispositivo móvil con sistemas operativos iOS y Android.

Dos compañeros de la parte de especialistas técnicos poseen un dispositivo paras estos requisitos, por lo cual hay que planificar su uso para las pruebas.

8. Riesgos

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

8.1. Planificación

Hay un gran riesgo por la parte de planificación que de momento está centralizada en el responsable de verificación. Se corre el riesgo de que muchos puntos importantes de la verificación queden sin asignar o con carga horaria insuficiente para poder generar lo que se solicita.

8.2. Técnico

Desde un punto de vista técnico, existe el riesgo de que tome demasiado tiempo la adecuación de los sistemas para correr los casos de prueba. La falta de experiencia en los lenguajes de codificación y en el uso de las herramientas puede generar un atraso importante en la realización de tareas de verificación.

8.3. Gestión

La gestión correcta de todos estos riesgos puede permitir mitigarlos, ya que organizar un equipo de hasta 4 personas que se ocupen de verificar el sistema puede brindan gran utilidad al uso de las horas/persona.

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.]