

Jogo Rubynho

Plan de Verificación y Validación

Versión 1.0

Historia de revisiones

Fecha	Semana	Versión	Descripción	Autor
29/08/14	3	1.0	Primera versión	Pablo Coore
13/09/14	5	1.1	Segunda version	Pablo Coore

Contenido

1. INTRODUCCIÓN	4
1.1. PROPÓSITO	4
1.2. PUNTO DE PARTIDA	4
1.3. ALCANCE	4
1.4. IDENTIFICACIÓN DEL PROYECTO.....	6
1.5. ESTRATEGIA DE EVOLUCIÓN DEL PLAN	6
2. REQUERIMIENTOS PARA VERIFICAR.....	6
3. ESTRATEGIA DE VERIFICACIÓN	7
3.1. TIPOS DE PRUEBAS	¡ERROR! MARCADOR NO DEFINIDO.
3.1.1. Prueba de integridad de los datos y la base de datos.....	7
3.1.2. Prueba de Funcionalidad	7
3.1.3. Prueba de Ciclo del Negocio.....	8
3.1.4. Prueba de Interfase de Usuario.....	9
3.1.5. Prueba de Performance.....	9
3.1.6. Prueba de Carga	10
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	12
3.1.10. Prueba de Fallas y Recuperación.....	13
3.1.11. Prueba de Configuración	14
3.1.12. Prueba de Instalación.....	15
3.1.13. Prueba de Documentos.....	15
3.2. HERRAMIENTAS	7
4. RECURSOS	16
4.1. ROLES	17
4.2. SISTEMA	17
5. HITOS DEL PROYECTO DE VERIFICACIÓN	18
6. ENTREGABLES	18
6.1. MODELO DE CASOS DE PRUEBA	18
6.2. INFORMES DE VERIFICACIÓN	18
6.3. EVALUACIÓN DE LA VERIFICACIÓN	19
6.4. INFORME FINAL DE VERIFICACIÓN	19
7. DEPENDENCIAS [OPCIONAL]	¡ERROR! MARCADOR NO DEFINIDO.
7.1. DEPENDENCIA DE PERSONAL [OPCIONAL]	¡ERROR! MARCADOR NO DEFINIDO.
7.2. DEPENDENCIA DE SOFTWARE [OPCIONAL].....	¡ERROR! MARCADOR NO DEFINIDO.
7.3. DEPENDENCIA DE HARDWARE [OPCIONAL]	¡ERROR! MARCADOR NO DEFINIDO.
7.4. DEPENDENCIA DE DATOS Y BASE DE DATOS DE PRUEBA [OPCIONAL]	¡ERROR! MARCADOR NO DEFINIDO.
8. RIESGOS [OPCIONAL]	20
8.1. PLANIFICACIÓN [OPCIONAL].....	20
8.2. TÉCNICO [OPCIONAL]	20
8.3. GESTIÓN [OPCIONAL]	20
9. APÉNDICE	21
9.1. NIVELES DE GRAVEDAD DE ERROR	21
9.2. NIVELES DE ACEPTACIÓN PARA LO ELEMENTOS VERIFICADOS	21

1. Introducción

1.1. Propósito

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

- *Verificar los prototipos realizados durante el proyecto así como todo entregable funcional que surja durante el proceso.*
- *Para cada versión o release del producto se verificará que se cumpla con las funcionalidades requeridas y los estándares de calidad y de codificación especificados por el cliente.*

Los recursos necesarios para realizar la verificación radican primeramente en los integrantes del grupo destinados a esta tarea que son:

- Pablo Coore: Responsable de verificación
- Mauricio Calcagno: Asistente de verificación
- Fernando Banchemo: Asistente de verificación
- Jorge Artave: Asistente de verificación
- Cristian Vega: Asistente de verificación

1.2. Punto de partida

Debido al enorme fanatismo que existe en Uruguay y en el mundo se ha decidido construir un proyecto que tiene como objetivo contar con un simulador Web de un juego de fútbol y una plataforma que permita definir estrategias de un equipo para retar y competir en ámbitos grupales e individuales.

Para lograr una verificación consistente, se marcaron como objetivos:

- Buscar una buena implementación de código, cumpliendo con los estándares, por parte de los implementadores, y los encargados de revisión a la hora de hacer pull request.
- Experimentar con distintas tecnologías de testing de modo que se tenga experiencia a la hora de aplicarlas al proyecto.
- Identificar componentes críticos del proyecto estableciendo una prioridad entre ellos para la verificación.
- Utilizar herramientas de software que contribuyen con la verificación.
- Orientar la verificación al cumplimiento de los estándares de calidad, además de cumplir los requerimientos.

1.3. Alcance

Se realizarán las siguientes actividades de verificación del proyecto:

Pruebas Unitarias:

Serán realizadas por los implementadores y es para asegurar el correcto funcionamiento de cada módulo. Se utilizará la herramienta de rails para realizar estas pruebas. Para evitar riesgos que afecten la implementación de la verificación se desarrolló una guía de cómo utilizar esta herramienta.

Pruebas de Integración:

Las pruebas de integración verifican que los componentes ya verificados unitariamente trabajen juntos como un sistema integrado. Estas también serán realizadas por los implementadores.

Revisión Documentos:

Los documentos requeridos por el cliente y los requeridos por los directores serán verificados para que sean legibles y entendibles, y cumplan con estándares determinados.

Revisión de código:

Se revisara el código implementado antes de subirlo al repositorio, de forma tal de reducir los errores y asegurar de que se cumplan los estándares de calidad de código. Esta tarea será realizada por el responsable de verificación y los asistentes.

Pruebas de Sistema:

Las pruebas del sistema verificarán el cumplimiento de los requerimientos funcionales y los no funcionales que hayan sido especificados, así como el correcto funcionamiento de las historias definidas. Estas pruebas serán realizadas por Pablo, Fernando y Cristian. Se utilizara la herramienta Cucumber descrita más adelante.

Los tipos de pruebas a realizar en un principio serian las siguientes:

- Integridad de los datos y la base de datos: Se realizarán pruebas para encontrar dicha integridad.
- Funcionalidad: Pruebas para verificar las funcionalidades que debe cumplir el sistema.
- Ciclo de Negocio: Pruebas para verificar ciclos de caso de uso comunes y asegurar su correcto funcionamiento.
- Interfaz de Usuario: Verifica que se cumplan las pautas definidas por el cliente.
- Performance: Se prueba el rendimiento del sistema
- Volumen: Se prueba el software ante grandes cantidades de datos para determinar si se alcanza algún límite de carga que cause fallas y manejarlo.
- Seguridad y Control de Acceso: Se buscará primordialmente seguridad en el ámbito del sistema, asegurando que sólo los usuarios habilitados para ingresar al mismo puedan hacerlo.
- Fallas y Recuperación: Probar la capacidad del software de recuperarse de alguna falla extrema, ya sea de software, hardware o de la red en sí.
- Instalación: La Prueba de Instalación buscará asegurar que el software pueda ser instalado en el ambiente objetivo y que una vez instalado

funcione correctamente (de acuerdo a las pruebas de funcionalidad, por ejemplo).

- Pruebas de configuración: se verificará que la aplicación web funcione para el o los navegadores indicados.

Aceptación:

Se aceptaran las pruebas si:

- Cumplen con el estándar de codificación
- Realizan una cobertura mayor al 90% del código
- Las pruebas se deben realizar en una maquina con: 4GB RAM, i7 y la aplicación debe poder soportar:
 - 15 partidos en simultáneo visualizándose. Armado de simulación y envío de info a viewers.
 - 10 viewers por partido (incluyendo a los jugadores)

Riesgos que podrían afectar la verificación:

- Dificultades debido a la falta de experiencia en las tecnologías utilizadas pueden provocar dificultad a la hora de encontrar defectos en el código.
- Problemas de comunicación entre los integrantes.

1.4. Identificación del proyecto

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

- Planes de Verificación y Validación de proyectos anteriores tomados de la memoria organizacional de la materia.
- Información recabada de internet.

1.5. Estrategia de evolución del Plan

- El responsable de monitorear el Plan de Verificación y Validación será el responsable de verificación Pablo Coore.
- Se modificara el plan a medida que cambien los requerimientos o surjan inconvenientes en el proyecto que afecten la verificación.
- Cualquier persona integrante del equipo de verificación podrá modificar el documento, y será evaluado por todos los integrantes del proyecto
- La comunicación de los cambios en el plan de verificación y validación será realizada por el correspondiente responsable a través de los medios ya estipulados.

2. Requerimientos para verificar

Requerimientos Funcionales:

Aun no se tienen requerimientos validados.

Requerimientos no funcionales:

- Cumplir con el estándar de codificación
- Utilización de Git/Github
- Manejo de historias o tareas a nivel de issues de Github

- Utilización de la herramienta de publicación de staging Heroku
- Debe poder ejecutarse en navegadores Firefox / Chrome / IE 9 o superior
- Alcanzar un %90 de cubrimiento con las pruebas unitarias
- Las pruebas se deben realizar en una máquina con: i7 con 4GB RAM
- La aplicación debe soportar: 15 partidos en simultáneo visualizándose. Armado de simulación y envío de info a viewers, 10 viewers por partido (incluyendo a los jugadores)

3. Estrategia de Verificación

Esta sección presenta el enfoque recomendado para la verificación. Describe como se verificarán los elementos.

Las estrategias definidas a continuación son solo de guía, las mismas pueden cambiar o eliminarse según las necesidades del cliente. Es por eso que puede que algunas pruebas no se ejecuten

Se indicarán las técnicas usadas y el criterio para saber cuando una prueba se completó (criterio de aceptación).

Las pruebas se deben ejecutar usando bases de datos conocidas y controladas en un ambiente seguro.

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 invoca cada método o proceso de acceso a la base de datos con datos válidos y no válidos.
- Se Inspecciona 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.

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 o historias 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 interface 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.

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

Se tiene que 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 simula 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 Interface de Usuario

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

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 Jogo Rubynho
- actualización, máquina previamente instalada con Jogo Rubynho, con la misma versión
- actualización, máquina previamente instalada con Jogo Rubynho, 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 Jogo Rubynho 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 Jogo Rubynho 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

Las herramientas que se utilizaran en el proyecto son:

- GoogleDrive: Repositorio donde se comparte información útil y documentos del proyecto.
- RubyMind: IDE para implementar el software y verificarlo.
- Git versión 2.1.0: Repositorio del proyecto de software.
- Heroku: Servidor donde correrán las aplicaciones.
- Rails versión 4.1.5: lenguaje de desarrollo web
- Ruby versión 2.1.2.p95: lenguaje de programación
- Rvm versión 1.25.29: Manejador de versiones de ruby
- Cucumber: framework para Ruby on rail, util por sus test de aceptacion automaticas permitiendo la ejecución de una carateristica del software escrita desde el punto de vista del usuario
- SimpleCov: gema que permite ver la cobertura del código
- PostgreSQL: Es un manejador de base de datos relacionales

Las siguientes herramientas serán investigadas para su utilización en el proyecto:

- QUnit: Herramienta que sirve para el testeo de los archivos de javascript

4. Recursos

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

Rol	Cantidad mínima de recursos recomendada	Responsabilidades
Responsable de verificación	15 hs	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	5hs	<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	--	<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	PostgreSQL
Red o subred	Internet
Nombre de la base de datos	A definir
Repositorio de pruebas	Servidor Github

Nombre del servidor	JogoRubynhoRepo
---------------------	-----------------

5. Hitos del Sprint del proyecto de Verificación

La verificación del Jogo Rubynho 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	6hs	29/08/14	06/09/14
Elaborar casos de prueba	10-20hs	03/09/14	06/09/14
Ajuste y Control de Verificación	5-10hs	03/09/14	13/09/14
Realización y ejecución de pruebas unitarias	20-30hs	15/09/14	27/09/14
Realización y ejecución de pruebas de sistema y de integración	20-30hs	22/09/14	17/09/14
Evaluar la verificación	2-4hs	15/09/14	28/09/14

6. Entregables

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

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

6.1. Modelo de Casos de Prueba

Documento	Modelo de Casos de Prueba
Creado por	El Responsable de verificación, Pablo Coore.
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 en este Sprint

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. Las fechas de liberación de las versiones de este documento serán realizadas en el próximo sprint (A partir del 15/09/2014).

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. Las fechas de liberación de las versiones de este documento serán realizadas en el próximo sprint (A partir del 15/09/2014).

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. Las fechas de liberación de las versiones de este documento están aun sin determinar

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. Las fechas de liberación de las versiones de este documento están aun sin determinar

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. Este documento será realizado en cada sprint

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. Riesgos

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

7.1. Planificación

Los riesgos de la planificación de la verificación son:

- Que la planificación sea de mala calidad impactando de manera negativa en la cantidad y calidad de las pruebas.
- Que el desarrollo de determinadas partes del software no se logren terminar a tiempo lo cual impactaría directamente en la planificación de la verificación.
- Que los integrantes del grupo tengan entregas o evaluaciones parciales de otras materias y no puedan dedicar el tiempo esperado al proyecto
- Que algunos de los integrantes se encuentren ausentes durante un periodo o todo el proyecto causando que el proyecto se retrase.

7.2. Técnico

El riesgo técnico de verificación es que el tiempo de aprendizaje de las herramientas de verificación lleve más tiempo de lo esperado.

7.3. Gestión

Para mitigar los riesgos que pueden surgir se sobreestimara el tiempo que lleva realizar las tareas y se tratara de comunicar los inconvenientes que pueden surgir al momento de ser encontrados.

8. Apéndice

8.1. 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.

8.2. 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.