**RubyTrick**

**Plan de Verificación y Validación**

**Versión 1.0**

**Historia de revisiones**

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Descripción | Autor |
| 28/08/2014 | 1.0 | Versión inicial | Grupo 3 |
| 31/08/2014 | 1.1 | Revisión de SQA | Santiago Gómez |
|  |  |  |  |
|  |  |  |  |

**Contenido**

[**1.INTRODUCCIÓN 3**](#h.gjdgxs)

[1.1.Propósito 3](#h.30j0zll)

[1.2.Punto de partida 3](#h.1fob9te)

[1.3.Alcance 3](#h.3znysh7)

[1.4.Identificación del proyecto 3](#h.2et92p0)

[1.5.Estrategia de evolución del Plan 3](#h.tyjcwt)

[**2.REQUERIMIENTOS PARA VERIFICAR 4**](#h.3dy6vkm)

[**3.ESTRATEGIA DE VERIFICACIÓN 4**](#h.1t3h5sf)

[3.1.Tipos de pruebas 5](#h.4d34og8)

[*3.1.1.Prueba de integridad de los datos y la base de datos 5*](#h.2s8eyo1)

[*3.1.2.Prueba de Funcionalidad 5*](#h.17dp8vu)

[*3.1.3.Prueba de Ciclo del Negocio 6*](#h.3rdcrjn)

[*3.1.4.Prueba de Interfaz de Usuario 6*](#h.26in1rg)

[*3.1.5.Prueba de Performance 7*](#h.lnxbz9)

[*3.1.6.Prueba de Carga 7*](#h.35nkun2)

[*3.1.7.Prueba de Esfuerzo (stress, competencia por recursos, bajos recursos) 8*](#h.1ksv4uv)

[*3.1.8.Prueba de Volumen 9*](#h.44sinio)

[*3.1.9.Prueba de Seguridad y Control de Acceso 9*](#h.2jxsxqh)

[*3.1.10.Prueba de Fallas y Recuperación 10*](#h.z337ya)

[*3.1.11.Prueba de Configuración 11*](#h.3j2qqm3)

[*3.1.12.Prueba de Instalación 11*](#h.1y810tw)

[*3.1.13.Prueba de Documentos 11*](#h.4i7ojhp)

[3.2.Herramientas 13](#h.2xcytpi)

[**4.RECURSOS 13**](#h.1ci93xb)

[4.1.Roles 13](#h.3whwml4)

[4.2.Sistema 13](#h.2bn6wsx)

[**5.HITOS DEL PROYECTO DE VERIFICACIÓN 14**](#h.qsh70q)

[**6.ENTREGABLES 14**](#h.3as4poj)

[6.1.Modelo de Casos de Prueba 14](#h.1pxezwc)

[6.2.Informes de Verificación 14](#h.49x2ik5)

[6.3.Evaluación de la verificación 15](#h.2p2csry)

[6.4.Informe final de verificación 15](#h.147n2zr)

[**7.APÉNDICE 17**](#h.3o7alnk)

[7.1.Niveles de gravedad de error 17](#h.23ckvvd)

[7.2.Niveles de aceptación para lo elementos verificados 17](#h.ihv636)

1. **Introducción**
   1. **Propósito**

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

* Identificar la información de proyecto existente y los componentes de software que deben ser verificados.
* 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. **Punto de partida**

Se verificará el sistema que será implementado por los desarrolladores del equipo Scrum. El mismo consistirá en un simulador web de fútbol, que por medio de lo lúdico ofrecerá una forma de aprender a programar en un lenguaje de programación amigable como lo es Ruby. Asimismo, este simulador también contará con una plataforma que permitirá a los usuarios registrarse, autenticarse, gestionar el equipo y programar su propia estrategia de juego en base a código, para luego competir con ella frente a otros usuarios mediante retos y competencias grupales.

* 1. **Alcance**

Se realizarán pruebas unitarias durante la etapa de construcción de cada una de las historias que serán desarrolladas, tanto de caja negra como de caja blanca. A medida que se completen las implementaciones de las historias, las mismas serán integradas en un ambiente de prueba (o staging) apropiado, para que se puedan llevar a cabo los tests de integración. Las pruebas en principio serán de carácter funcional, y se encargarán de verificar cada uno de los requerimientos especificados. En etapas finales del desarrollo, se llevarán a cabo pruebas de performance, con el objetivo de probar cuánta carga de datos o de usuarios es capaz de soportar el sistema.

Por el momento no se cuenta con el backlog definido en su totalidad, por lo que este plan de verificación podría sufrir modificaciones a lo largo del proyecto. También es sabido que, al trabajar en metodologías ágiles, los cambios son naturalmente aceptados, y por lo tanto podrían acontecer cambios importantes en los requerimientos funcionales o no funcionales, que impliquen como consecuencia obvia una modificación correspondiente en el plan de verificación.

* 1. **Identificación del proyecto**

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

* Documento de requerimientos (o backlog).
  1. **Estrategia de evolución del Plan**
* El equipo Scrum en su conjunto será el encargado de monitorear el plan descrito en este documento.
* Se esperan realizar modificaciones del presente documento con una frecuencia de una vez por cada iteración o de una vez por semana, dependiendo de los cambios que acontezcan en el transcurso del proyecto.
* Todo cambio será debidamente analizado por el equipo Scrum en su totalidad, y en consecuencia, se determinará en conjunto el cambio a realizar en el plan de verificación.
* Se generará una nueva versión del documento, con fecha y motivo de cambio debidamente detalladas en el mismo, y dicho cambio será comunicado tanto al equipo de desarrollo como al cliente y director de proyecto, por el medio de comunicación que corresponda.

1. **Requerimientos para verificar**

La lista de requerimientos funcionales y no funcionales a ser verificados no se encuentra aún determinada, dado que el proceso de relevamiento de requisitos aún no culminó, y no se cuenta todavía con el backlog del proyecto definido y detallado, por lo que la siguiente lista de requisitos describe de manera muy vaga los aspectos a verificar, y estará por ende, sujeta a modificaciones en el correr de las próximas semanas. Se espera verificar a grandes rasgos, las siguientes historias:

* Simular Partido.
* Gestión de usuarios
  + Registro, inicio y cierre de sesión. Modificar y ver datos de usuario.
* Gestión de Amigos
  + Agregar, borrar y aceptar.
* Gestión de Jugadores
  + Ver, vender, comprar, entrenar y despedir.
* Gestión de equipo y sus estrategias
  + Ver equipo, ver, agregar y borrar estrategias de equipo.
* Gestión de Campeonatos
  + Unirse y crear campeonatos. Ver datos de campeonatos: fixture y resultados.
* Página de Inicio.
* Sección de Notificaciones del usuario.
* Tutorial de “Cómo jugar”, que enseña a utilizar la plataforma.
* Motor del juego
  + Jugador, pelota, partido, cancha.

1. **Estrategia de Verificación**

Se realizarán pruebas unitarias, tanto de caja negra como de caja blanca, de cada módulo que se implemente en el sistema. Para ello se utilizará como herramienta a Rspec, una herramienta de testing para el lenguaje Ruby, dado que será fácil de instalar y de integrar con las demás herramientas.

Respecto a las pruebas de caja negra, la técnica a abordar será la de partición en clases de equivalencia, en donde se intentará probar con conjuntos de datos de datos de entrada que representen a cada una de las distintas clases de equivalencia que existan. Además, se probará también con valores límite. Para ello, se diseñarán pruebas de manera tal de considerar estas clases de datos de entrada.

En las pruebas de caja blanca, se aplicará la técnica de cubrimiento de sentencias, en donde se planteará como objetivo mínimo lograr un cubrimiento del 90 % del total de sentencias de cada módulo a implementar.

No se llevarán a cabo técnicas de cubrimiento de decisión, ni de decisión/condición, dado que las mismas implcarían evaluar cada una de las combinaciones posibles de valores de cada condición en cada estructura de control del código, lo que puede insumir un costo en el esfuerzo de testing que puede llevar a retrasar el proyecto, teniendo como consecuencia el no cumplimiento de los plazos establecidos para la liberación del producto.

En cuanto al criterio de detención de las pruebas de caja negra, el mismo será el haber realizado todas las pruebas diseñadas, y detectados y corregidos todos los errores que las pruebas de caja negra generaron.

* 1. **Tipos de pruebas**

* + 1. **Prueba de integridad de los datos y la base de datos**
       1. *Objetivo de la prueba*

Asegurar que los métodos y procesos de acceso a la base de datos funcionan correctamente y sin corromper datos.

* + - 1. *Técnica*

Invoque cada método o proceso de acceso a la base de datos con datos válidos y no válidos.

Inspeccione la base de datos para asegurarse de que se han guardado los datos correctos, que todos los eventos de la base de datos ocurrieron correctamente, o repase los datos devueltos para asegurar que se recuperaron datos correctos por la vía correcta.

* + - 1. *Criterio de aceptación*

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

* + - 1. *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.

* + 1. **Prueba de Funcionalidad**
       1. *Objetivo de la prueba*

Asegurar el correcto funcionamiento de cada una de las historias o funcionalidades del sistema.

* + - 1. *Técnica*

Ejecute cada 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.
  + - 1. *Criterio de aceptación*

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

* + - 1. *Consideraciones especiales*

No hay consideraciones especiales.

* + 1. **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.

* + - 1. *Objetivo de la prueba*

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

* + - 1. *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.
  + - 1. *Criterio de aceptación*

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

* + - 1. *Consideraciones especiales*

No hay consideraciones especiales.

* + 1. **Prueba de Interfaz de Usuario**
       1. *Objetivo de la prueba*

Verificar que: la navegación a través de las páginas HTML que se están probando reflejen las funciones del negocio y los requerimientos, incluyendo manejo de botones, campos y links; los objetos de los formularios y características, como menúes, tamaño, posición, estado funcionen de acuerdo a los estándares.

* + - 1. *Técnica*

Establecer criterios para determinar cuándo una página o formulario HTML se considera “correcta”, en el sentido de que satisface los requerimientos establecidos para las interfaces de usuario.

* + - 1. *Criterio de aceptación*

Cada formulario o página del sistema ha sido verificada exitosamente siendo consistente con una versión de referencia o estándar establecido.

* + - 1. *Consideraciones especiales*

No hay consideraciones especiales.

* + 1. **Prueba de Performance**
       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.
  + - 1. *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).
  + - 1. *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.

* + - 1. *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 usuarios, generalmente varios cientos.
* 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.

* + 1. **Prueba de Carga**
       1. *Objetivo de la prueba*

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

* + - 1. *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.

* + - 1. *Criterio de aceptación*

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

* + - 1. *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.

* + 1. **Prueba de Esfuerzo (stress, competencia por recursos, bajos recursos)**
       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.

* + - 1. *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.

* + - 1. *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.

* + - 1. *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.

* + 1. **Prueba de Volumen**
       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.
  + - 1. *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.
  + - 1. *Criterio de aceptación*

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

* + - 1. *Consideraciones especiales*

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

* + 1. **Prueba de Seguridad y Control de Acceso**
       1. *Objetivo de la prueba*

Seguridad en el ámbito de aplicación: Verificar que un usuario pueda acceder solo a las funciones o datos para los cuales su tipo de usuario tiene permiso. Verificar en concreto que un usuario común y corriente no pueda eliminar otras cuentas de usuario a menos que este sea el administrador del sistema.

Seguridad en el ámbito de sistema: Verificar que solo los usuarios con acceso al sistema y a las aplicaciones, puedan acceder a ellos. Verficar en concreto, que sólo los usuarios registrados en el sistema tengan acceso a las funcionalidades que el mismo ofrece.

* + - 1. *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.

* + - 1. *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.

* + - 1. *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.

* + 1. **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.

* + - 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.
  + - 1. *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.
  + - 1. *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.

* + - 1. *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.

* + 1. **Prueba de Configuración**

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

* + - 1. *Objetivo de la prueba*

Verificar que el software funcione apropiadamente en los siguientes navegadores web: Firefox, Chrome y IE versión 9, o superior.

* + - 1. *Técnica*

Usar las pruebas de Funcionalidad.

* Abrir y cerrar varias sesiones de usuario utilizando estos tres navegadores distintos.
* Ejecutar varias operaciones en cada una de las sesiones de usuario y en cada navegador.
  + - 1. *Criterio de aceptación*

Todas las operaciones fueron completadas sin fallas para cada funcionalidad que ha sido probada en cada uno de estos tres navegadores mencionados en el objetivo de la prueba.

* + - 1. *Consideraciones especiales*

No hay consideraciones especiales.

* + 1. **Prueba de Instalación**

Dado que el producto a desarrollar operará como un servicio web, no será necesario ningún proceso de pruebas de instalación para su uso, siendo únicamente necesario ejecutar un browser o navegador web (Chrome, Mozilla o IE).

* + 1. **Prueba de Documentos**
       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.
  + - 1. *Técnica*

Se hará una revisión al documento del plan de pruebas (específicamente la sección de escenarios-condición) previa a la generación de los casos de prueba de integración y funcionales basados en historias, con el fin de verificar que el documento contiene toda la información necesaria para la generación de datos de prueba y que ésta es correcta; esto es, concluir que se tienen en cuenta todos los escenarios-condición, verificar que las salidas son correctas según la especificación de las historias y verificar que las entradas y salidas están correctamente especificadas (no son ambiguas, se corresponden con el diseño del sistema y son comprensibles para el equipo encargado de generar los casos de prueba).

Se hará también una revisión del documento de requerimientos para verificar que los requerimientos especificados sean verificables o en su defecto sean lo más específicos posible (esto es que no sean ambiguos, sean comprensibles para el equipo de desarrollo).

También se revisará el documento de gestión de riesgos para verificar principalmente la correcta clasificación de los riesgos (tipo, probabilidad, impacto y exposición).

Las revisiones son realizadas por ciertos integrantes del equipo scrum, en presencia de los autores del documento y en el caso del documento de requerimientos también asiste por lo menos un representante del equipo scrum. Se confecciona una lista de todos los aspectos a verificar de cada documento y se considera que un documento pasó la verificación si contiene todos los aspectos listados.

Para las inspecciones de código, dos integrantes del equipo se encargarán de inspeccionar cada fuente que se codifique y se solicite subir mediante un pull request, y una vez que los mismos hayan sido debidamente revisados y aprobados por ambos miembros, se realizará el commit en el repositorio. Al igual que con la documentación, existirá una lista de ítems o aspectos que los archivos fuente deberán cumplir para poder pasar la prueba de inspección, y que serán objeto de revisión por parte de estos miembros del equipo.

* + - 1. *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.

* + - 1. *Consideraciones especiales*

No hay consideraciones especiales.

* 1. **Herramientas**

Rspec – Testing unitario y de integración – Open Source.

Git – Control de versiones – Open Source

Github – Repositorio del proyecto – Open Source

Heroku – Ambiente para el deployment – Propiedad de Salesforce.com

Pgadmin – Administrador de bases de datos PostgreSQL – Open Source

1. **Recursos**
   1. **Roles**

En la tabla a continuación se muestra la composición de personal para el proyecto RubyTrick 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 | 9 | * 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. |

* 1. **Sistema**

|  |  |
| --- | --- |
| **Recurso** | **Nombre/Tipo** |
| Servidor de base de datos | PostgreSQL |
| Nombre del servidor | ---- |
| Nombre de la base de datos | ---- |
| PC Cliente para pruebas | ---- |
| Requerimientos especiales | ---- |
| Repositorio de pruebas | ---- |
| Nombre del servidor | ---- |

1. **Hitos del proyecto de Verificación**

La verificación del RubyTrick 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 |  |  |  |
| Elaborar casos de prueba |  |  |  |
| Ajuste y Control de Verificación |  |  |  |
| Ejecutar la verificación |  |  |  |
| Evaluar la verificación |  |  |  |

1. **Entregables**
   1. **Modelo de Casos de Prueba**

|  |  |
| --- | --- |
| Documento | **Modelo de Casos de Prueba** |
| Creado por | El Responsable de verificación, Pablo Carballo. |
| 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 (a definir). |

* 1. **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.  Versión:  Fecha: |

|  |  |
| --- | --- |
| 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.  Versión:  Fecha: |

|  |  |
| --- | --- |
| 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.  Versión:  Fecha: |

|  |  |
| --- | --- |
| 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. Fecha: |

* 1. **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.  Fecha: |

* 1. **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. |

1. **Apéndice**
   1. **Niveles 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.
  1. **Niveles de aceptación para lo elementos verificados**
* **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.]