ASH Web

Plan de Verificación y Validación

Versión 1.2

Historia de revisiones

Fecha	Versión	Descripción	Autor
25/08/2016	1.0	Creación del Documento	Rodrigo Viera
28/08/2016	1.1	Revisión de SQA	Isabel Ivagnes
01/09/2016	1.2	Actualización del Doc.	Rodrigo Viera
04/09/2016	1.3	Revisión de SQA	Alvaro Callero

Contenido

1.Introducción

1.1. Propósito

Este Plan de Verificación para el proyecto ASH (Animales Sin Hogar) soporta los siguientes objetivos:

- Identificar la información del proyecto y los componentes de software que deben ser verificados.
- Enumerar las historias para verificar. En este documento se hace referencia a historia y no a requerimiento, ya que se utiliza SCRUM como metodología de desarrollo.
- Describir las estrategias de verificación que se utilizarán.
- Identificar los recursos necesarios y proporcionar una estimación de esfuerzo para realizar la verificación.
- Enumerar los entregables del proyecto a verificar.

1.2. Punto de partida

El proyecto presentado por TopTier Labs tiene como objetivo desarrollar una API y una aplicación Web que consuma los métodos que ofrezca la API. La aplicación Web deberá permitir que los usuarios del sistema ASH puedan registrar los animales con los que ya cuenta el refugio, ingresar nuevos animales que lleguen al refugio, crear y controlar las adopciones, ingresar adoptantes con su respectiva información, asignar permisos a determinados usuarios, registrar eventos que involucren a los animales (ej: intervenciones médicas), filtrado de información y exportación de la información que se desee.

La Verificación tiene como objetivo poder entregarle al cliente un producto de calidad, buscando minimizar las probabilidades de que el sistema falle cuando se encuentre en producción. La Verificación de este sistema buscará descubrir defectos que puedan ocurrir durante el desarrollo del proyecto por parte del equipo de desarrolladores y así lograr un correcto funcionamiento del producto en base a lo acordado con el cliente.

Se deberá verificar que todas las historias y requerimientos no funcionales especificados por el cliente sean cumplidos para lograr la satisfacción del mismo.

1.3 Alcance

Se utilizará la gema RSpec como framework de Testing para el Back-end (RoR), MochaJS como framework de Testing para el Front-end (React) y Travis CI para las Pruebas de Integración Continua.

Cuando las historias se terminen de implementar, estas serán notificadas para su Verificación.

En el presente proyecto se intentará contar con las siguientes fases de Verificación:

1. ***Pruebas Unitarias:*** Estas pruebas tienen como objetivo verificar las funcionalidades de cada módulo. Los encargados de realizar estas pruebas son los desarrolladores que implementaron el módulo en conjunto con los verificadores siempre que se lo requiera.

Es importante que luego de realizar estas pruebas, los desarrolladores realicen informes de verificación unitaria conteniendo la información de los errores encontrados y especificando si estos pudieron ser resueltos.

Por último, luego de que los desarrolladores ejecutaron las pruebas unitarias, el equipo de Verificación deberá validar que los módulos han sido correctamente verificados y en caso de que algún módulo no se haya verificado de forma correcta, se deberán de realizar las pruebas que se consideren convenientes.

Estas pruebas serán ejecutadas para cada Sprint siempre que se implementen nuevas funcionalidades sobre el sistema.

2. **Pruebas de Integración**: Estas pruebas tienen como objetivo verificar que los componentes interactúen correctamente y se realizan una vez que se han aprobado las pruebas unitarias. Los encargados de realizar estas pruebas son los desarrolladores que implementaron los módulos en conjunto con los verificadores. Estas pruebas serán ejecutadas siempre que se finalice la implementación de diferentes módulos.

3. ***Pruebas Funcionales:*** Estas pruebas tienen como objetivo verificar que el sistema realice correctamente las historias. El encargado de realizar estas pruebas es el equipo de Verificación. Estas pruebas serán ejecutadas para cada Sprint siempre que se realicen nuevas funcionalidades.

4. ***Pruebas No Funcionales:*** Estas pruebas tienen como objetivo verificar que el sistema cumpla con los requerimientos no funcionales. Por ejemplo: la interfaz de Usuario deberá ser sencilla y de fácil aprendizaje, la app Web deberá funcionar en los Navegadores Chrome (v54), Firefox (v48) e IE (v12), etc. El encargado de realizar estas pruebas es el equipo de Verificación.

5. ***Pruebas de Aceptación:*** Estas pruebas tienen como objetivo verificar que el sistema cumple con lo solicitado por el cliente. Estas pruebas se realizan al finalizar cada Sprint. El encargado de realizar estas pruebas es el equipo de Verificación en conjunto con el Cliente.

1.4. Identificación del proyecto

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

- Documento de Especificación de Requerimientos
- Backlog de Trello (Gestor de tareas)

1.5. Estrategia de evolución del Plan

Para cada Sprint se definirá con el Cliente que pruebas se realizarán ya que se está trabajando con metodología SCRUM. Para el Sprint 1 solicitó ejecutar pruebas unitarias para cada controlador que se haya desarrollado en el sistema y que estas ejecuten satisfactoriamente.

Como se expresó en la parte anterior, la verificación se divide en tres etapas. En primer lugar se encuentran las pruebas unitarias. En esta etapa los desarrolladores deberán generar y

correr los casos de prueba, para luego documentar los resultados y enviarlos al equipo de verificación.

En segundo lugar se encuentran las pruebas de integración. Estas pruebas también son diseñadas por los desarrolladores, quienes luego de correr las pruebas también deben documentar los resultados y enviarlos al equipo de verificación.

Aunque los desarrolladores son los encargados de estas tareas debido a su mayor conocimiento y capacidad de interpretación, los verificadores se encontrarán a su disposición para auxiliarlos y en caso de ser necesario, para complementar y crear nuevos casos de prueba.

En tercer lugar se encuentran las pruebas del sistema y están a cargo del equipo de verificación. Para esto se definen las siguientes etapas:

● Diseño de pruebas: Se lleva a cabo por el Responsable de Verificación con la colaboración del equipo de verificación. Para mitigar riesgos se comienza de forma paralela a la implementación

● Validación de las pruebas: El equipo de verificación debe juzgar como correcta cada prueba antes de ejecutarla

● Preparación del ambiente de pruebas: Se debe concebir un ambiente de pruebas desacoplado del ambiente de desarrollo para realizar las pruebas del sistema. Este ambiente será concebido en conjunto por el Responsable de SCM y los verificadores, pudiendo contar con la ayuda de especialistas técnicos si fuese necesario.

● Ejecución de las pruebas: Realizado por el equipo de verificadores.

● Conclusión de las pruebas: Las pruebas pueden ser aprobadas o no. También pueden ser aprobadas con observaciones. En cualquier caso se realiza un informe de la prueba y se lleva a cabo una devolución al equipo de desarrollo

Otras aclaraciones sobre el plan de Verificación y Validación:

- El responsable de monitorear el Plan de Verificación y Validación es el responsable de Verificación.
- Todo cambio en el plan será analizado por el equipo de Verificación para determinar si es necesario realizarlo.
- El plan será actualizado por el equipo de Verificación siempre que el proyecto lo demande con previa autorización del responsable de Verificación.
- Se creará una nueva versión del documento con la fecha y motivo del cambio siempre que se apruebe un cambio y se le informará al equipo de desarrollo y al cliente por el medio de comunicación que corresponda.
- Se tendrá una copia del documento en GitHub para que el cliente pueda acceder a la misma siempre que lo desee y otra copia en Google Drive a disposición del equipo.

2. Requerimientos para verificar

Historias a verificar:

1. Como Usuario del Sistema: (Sprint 1)

Creación de cuenta de voluntario.

2. Como Administrador del Sistema:

Aceptar o rechazar solicitudes de creación de cuenta. (Sprint 1)

Activar cuentas.

El sistema deberá enviar un mail al Usuario correspondiente a la cuenta aceptada.

3. Como voluntario: (Sprint 1)

Realizar login en el sistema.

4. Como Administrador:

Asignar y quitar permisos a los usuarios.

5. Como Voluntario con permiso An:

Crear animales

Ingresar un animal en el sistema.

6. Como Voluntario con permiso An:

Modificar la información de un animal.

7. Como Voluntario con permiso An:

Registrar un evento para un animal

Agregar fotos al evento.

8. Como Voluntario:

Realizar búsqueda de animales teniendo la posibilidad de usar filtros según los distintos atributos de los mismos.

9. Como Voluntario:

Exportar los resultados de una búsqueda a Excel o PDF.

10. Como voluntario:

Exportar la lista de eventos de un único animal a Excel o PDF.

11. Como voluntario con permiso Ad:

Crear un adoptante

Vincular Adoptante con un animal

Agregar Adoptante a la blacklist y agregar comentarios

Vincular animal con adoptante

Actualizar el estado de adopción de un animal

12. Como voluntario:

Consultar estadísticas sobre las adopciones de los animales.

13. Como voluntario:

Obtener estadísticas sobre las blacklists de los animales.

Requisitos No Funcionales a verificar:

1. Se deberá utilizar Ruby on Rails para el desarrollo del back-end, React para el desarrollo del front-end, Postgresql como motor de base de datos, RSpec para los tests de back-end y MochaJS para los tests de front-end.

2. La aplicación web debe ser compatible y funcionar correctamente con los navegadores web Firefox (48), Chrome (54) e IE en su última versión (12).

3. La interfaz de Usuario deberá ser sencilla para el fácil aprendizaje y manejo por parte de los usuarios. Se deberá tener en cuenta al momento de su diseño los mockups de las vista y pantallas ofrecidas por el cliente.

3. Estrategia de Verificación

3.1. Tipos de pruebas

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

Objetivo de la prueba

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

Técnica

Se invocará a cada método o proceso de acceso a la base de datos con datos válidos y no válidos. Posterior a la invocación, se accederá a 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 y que no se generaron datos erróneos.

Criterio de aceptación

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

Consideraciones especiales

La prueba requiere un entorno de administración de DBMS o controladores para ingresar o modificar información directamente en la base de datos. Se utilizará PgAdmin III sobre PostgreSQL.

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.

Prueba de Funcionalidad

Objetivo de la prueba

Asegurar el correcto funcionamiento de cada historia y de cada funcionalidad del sistema.

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.

Estas pruebas serán realizadas por los verificadores en el transcurso de cada sprint mientras se desarrollan nuevas funcionalidades.

Criterio de aceptación

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

Consideraciones especiales

No aplica.

Prueba de Interfaz de Usuario

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 links, botones, y campos; los objetos de los formularios y características, como menúes, tamaño, posición, estado funcionen de acuerdo a los estándares y que sean limpios y claros para brindar un uso fluido y fácil aprendizaje.

Técnica

Establecer criterios para determinar cuándo una página se considera “correcta”, en el sentido de que satisface los requerimientos establecidos para las interfaces de usuario. Estas pruebas serán realizadas por los implementadores de front-end y los verificadores en caso de ser necesario, en el transcurso del sprint mientras se desarrollan nuevas funcionalidades. Siempre que se encuentren errores estos serán notificados a los verificadores.

Criterio de aceptación

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

Consideraciones especiales

No aplica.

Prueba de Performance

No aplica. Se entiende que la aplicación en una primera instancia solamente será utilizada de forma interna por pocos usuarios. Sin embargo, se tendrá en cuenta que la aplicación ejecute sin demoras significativas luego de realizar una transacción genérica.

Prueba de Carga

No aplica. Se entiende que la aplicación en una primera instancia solamente será utilizada de forma interna por pocos usuarios y que no se tendrá excesiva carga de datos ni transacciones muy demandantes.

Prueba de Esfuerzo

No aplica. Se entiende que la aplicación en una primera instancia solamente será utilizada de forma interna por pocos usuarios.

Prueba de Volumen

No aplica. Se entiende que la aplicación en una primera instancia solamente será utilizada de forma interna y no se cuenta con una cantidad excesiva de usuarios que hagan uso del sistema y de funcionalidades al mismo tiempo.

Prueba de Seguridad y Control de Acceso

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.

Seguridad en el ámbito de sistema: Verificar que solo los usuarios con acceso al sistema y a las aplicaciones, puedan acceder a ellos. Es decir, verificar que sólo los usuarios registrados en el sistema tienen acceso a las funcionalidades que el mismo ofrece. Se comprobará que no existen datos guardados en la base de datos en formato de texto plano y que estén encriptados.

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.

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

Criterio de aceptación

Para cada tipo de usuario 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.

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.

Prueba de Configuración

Objetivo de la prueba

Verificar que el software funcione apropiadamente en los navegadores web: Chrome (v54), Firefox (v48) e IE (v12).

Técnica

Usar las pruebas de Funcionalidad.

Abrir y cerrar varias sesiones de usuario utilizando los navegadores antes mencionados.

Ejecutar varias operaciones en cada una de las sesiones de usuario y en cada navegador.

Estas pruebas serán realizadas por los implementadores y los verificadores en el transcurso del sprint mientras se desarrollan nuevas funcionalidades. Al final de cada sprint se volverán a ejecutar todas las pruebas para verificar que no hayan ocurrido errores y en caso de haber puedan ser corregidos antes de la demo con el cliente.

Criterio de aceptación

Todas las operaciones son completadas exitosamente sin fallas para cada funcionalidad que ha sido probada en cada uno de los tres navegadores mencionados.

Consideraciones especiales

Se recomienda realizar Testing Exploratorio.

Prueba de Instalación

No aplica. El sistema desarrollar operará como un servicio Web.

Prueba de Documentos

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.

Técnica

En conjunto con el Responsable de SQA se revisará el documento de plan de pruebas antes de la generación de los casos de pruebas funcionales basados en las historias, con el objetivo de verificar que se tiene toda la información necesaria para la generación de datos de prueba y que ésta sea correcta, es decir, que se cuenta con todos los escenarios-condición, que las salidas son correctas según la especificación de las historias y que las entradas/salidas no son ambiguas y son comprensibles.

Se revisará el documento de requerimientos para verificar que los requerimientos especificados sean verificables o en su defecto sean lo más específicos posibles.

Se revisará el documento de gestión de riesgos para verificar principalmente la correcta clasificación de los riesgos.

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.

Consideraciones especiales

No aplica.

3.2 Herramientas

- Ruby on Rails - Lenguaje de programación para Back-end.
- React - Lenguaje de programación para Front-end.
- Git - Control de versiones
- GitHub - Repositorio del proyecto
- RSpec - Testing Unitario y de Integración
- MochaJS - Testing Unitario React
- Travis CI - Integración continua

- Heroku - Ambiente para deployment
- PgAdmin III - DBMS PostgreSQL
- Slack - Comunicación entre miembros del equipo
- Trello - Gestor de tareas y de bugs

4. Recursos

4.1. Roles

En la tabla a continuación se muestra la composición de personal para el proyecto ASH 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	4	Ejecuta las pruebas. Registra los resultados de las pruebas. Recupera el software de errores. Documenta los pedidos de cambio.

4.2. Sistema

Recurso	Nombre/Tipo
Servidor de base de datos	PostgreSQL
PC Cliente para pruebas	PC con Linux
Repositorio de pruebas	GitHub

5. Hitos del proyecto de Verificación

La verificación del ASH 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	8 hs	25/08/16	27/08/16
Elaborar casos de prueba	Sin definir.	-	-
Ajuste y Control de Verificación	Sin definir.	-	-
Ejecutar la verificación	Sin definir.	-	-
Evaluar la verificación	Sin definir.	-	-

6. Entregables

6.1. Modelo de Casos de Prueba

Documento	Archivo de Excel con casos de prueba.
Creado por	El Responsable de verificación, Rodrigo Viera.
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 en cada Sprint.
Fecha de liberación	Será liberado durante el sprint 2.

6.2. Informes de Verificación

Documento	Se genera un documento *Informe de Verificación Unitaria* por cada prueba unitaria que se realice al sistema.
Creado por	Las personas que ejecutan las pruebas.
Para quien	Es el retorno para los implementadores de la tarea de verificación, que detalla los errores encontrados para que puedan ser corregidos.
Fecha de liberación	Será liberado luego de cada verificación unitaria.

Documento	Se genera un documento **Informe Consolidación** por cada consolidación que se realice al sistema.
Creado por	Las personas que ejecutan las pruebas.
Para quien	Es el retorno para los implementadores de la tarea de consolidación, que detalla los errores encontrados para que puedan ser corregidos.
Fecha de liberación	Será liberado luego de cada consolidación.

Documento	Se genera un documento ***Informe de Verificación de Integración*** por cada prueba de integración que se realice al sistema.
Creado por	Las personas que ejecutan las pruebas de integración.
Para quien	Es el retorno para los implementadores de la tarea de verificación, que detalla los errores encontrados para que puedan ser corregidos.
Fecha de liberación	Será liberado luego de cada verificación de integración.

Documento	Se genera un documento ***Informe de Verificación de Sistema*** por cada prueba de sistema que se realice.
Creado por	Las personas que ejecutan las pruebas.
Para quien	Es el retorno para los implementadores de la tarea de Verificación, que detalla los errores encontrados para que puedan ser corregidos.
Fecha de liberación	Será liberado luego de cada verificación del sistema.

6.3. Evaluación de la verificación

Documento	El documento *Evaluación de la verificación* será creado por cada prueba que se realice al sistema. Contiene las fallas encontradas en el sistema, la cobertura de la verificación realizada y el estado del sistema.
Creado por	Responsable de Verificación, Rodrigo Viera.
Para quien	Es el resumen de la tarea de verificación y es el retorno para todo el equipo de trabajo del estado del sistema.
Fecha de liberación	Será liberado luego de cada verificación, unitaria, de integración y de sistema.

6.4. Informe final de verificación

Documento	El documento *Informe final de verificación* es el resumen de la verificación final del sistema antes de que sea liberado al entorno del usuario.

Creado por	Responsable de Verificación, Rodrigo Viera
Para quien	Desarrolladores, Cliente, Director de Proyecto
Fecha de liberación	Será liberado luego de la verificación final del sistema.

7. Riesgos

7.1. Planificación

Un posible riesgo es cometer errores en la planificación del proyecto, es decir, en el manejo de las estimaciones para cada tarea, ya que si el desarrollo de una tarea consume más tiempo que el tiempo que se le estimó, podría afectar el tiempo dedicado para la Verificación e impactar sobre la calidad de las pruebas y la calidad del proyecto y por ende podría no cumplirse el plan de Verificación y Validación.

7.2. Gestión

Para mitigar posibles riesgos en las tareas de Verificación, se intentará sobreestimar el tiempo necesario de desarrollo de cada tarea, de forma de tener un "colchón" que permita no atrasarse con las entregas de cada sprint.

8. Apéndice

8.1. 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.

8.2. 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.