Plan de administración de configuraciones

- Control de Versiones
    - Version 0.1.0 - Autor: Kleiner Matias - Fecha: 03/04/2017 - Descripción: primera versión del documento
    - Versión 0.1.1 - Autor: Cavanagh Juan - Fecha: 06/05/2017 - Descripción: corrección de ortografía e incorporación de secciones.
    - Versión 0.1.2 - Autor: Kleiner Matias - Fecha: 06/05/2017 - Descripcion: Imagenes de esquemas de ramas y directorios agregadas.
    - Versión 0.2.0 - Autor: Cavanagh Juan - Fecha: 07/05/2017 - Descripción: modificación de la sección administración de Release, corrección de errores en diferentes secciones.
    - Versión 0.2.1 - Autor: Kleiner Matias - Fecha: 27/05/2017 - Descripción: cambio de extension del documento (.md a pdf)

1. Introducción

Este documento describe el plan de administración de la configuración para el proyecto final de la materia Ingeniería de Software. El proceso de Administración de la Configuración asegura el control de las distintas versiones de los documentos entregables, los responsables de cada uno, el tiempo y los recursos generados durante el ciclo de vida del proyecto.

1.1 Propósito

El propósito de este documento es establecer los elementos necesarios para administrar los documentos y las fuentes que son elaborados por el equipo del proyecto.

- Coordinar el uso y actualización de los módulos que se encuentren en línea base del proyecto.
- Asegurar que todos estén trabajando en las mismas versiones de cambios.
- Controlar que ninguno de estos cambios se pierda.
- Proveer criterios y direcciones necesarias para el desempeño del proceso de administración de la configuración.
- Decidir si es conveniente o no implementar un cambio solicitado.
- Especificar roles de cada participante del proyecto.
- Definir cómo se realizará el backup del proyecto.

1.2 Glosario

Los siguientes términos serán frecuentemente mencionados a lo largo del documento por lo tanto se considera necesario dejar en evidencia sus definiciones:

Repositorio: Lugar donde se almacenan los datos actualizados e históricos en un servidor. Puede ser un sistema de archivos en un disco duro, un banco de datos, etc.

Rama-Módulo (“Branch”): Conjunto de directorios y/o archivos dentro del repositorio que pertenecen a un proyecto común.

Revisión ("versión"): Una revisión es una versión determinada de un archivo.

Release: Lanzamiento del producto, puesta en venta.

Cambio: Representa una modificación específica a un documento bajo control de versiones.

Comité de Control de Cambios (“Change Control Board”): Grupo de personas responsables de evaluar y aprobar determinados cambios propuestos sobre un ítem de configuración. Además verifican la implementación de los cambios aprobados.

Build: Generar compilar y/o ejecutar un sistema.

Commit: Comentario que describe brevemente los cambios realizados en el repositorio.

Línea Base: Conjunto de productos de trabajo que han sido revisados y aprobados, y que serán utilizados luego como base para la realización de cambios. Dichos cambios sólo podrán realizarse a través de un proceso formal de control de cambios.

S x.x: Corresponde a la sección determinada según los valores de x. El lector deberá dirigirse a esa sección para obtener más informacion (Ej: S 1.3 corresponde a la Sección 1.3 Herramientas de administración de configuraciones)

1.3 Herramientas de la administración de configuraciones

Las herramientas a continuación serán utilizadas para facilitar el desarrollo del proyecto y asegurar una correcta integración del trabajo de los desarrolladores.

- Herramienta de control de versiones (GitHub): Se creará un repositorio en el servidor con el nombre del proyecto, al cual podrán acceder todos los miembros del equipo de desarrollo como colaboradores. (*BITSoftware*)
- Herramienta de integración contínua (TravisCI/Gradle): A medida que los desarrolladores modifiquen algún ítem de configuración, esta herramienta correrá automáticamente los test corroborando que no existan errores. En caso de errores la herramienta se encargará de notificar a todos los desarrolladores.
- Herramienta de control de defectos (Issues/GitHub): Se utilizará la herramienta de control de defecto para tener seguimiento de los errores e inconvenientes surgidos en el proyecto BITSoftwareIssues

2. Administración de la configuración

2.1 Roles y responsabilidades

Las actividades de la gestión de las configuraciones dentro del proyecto serán coordinadas por 3 integrantes, los cuales se encuentran en el mismo nivel jerárquico. Sin embargo cada uno posee funciones específicas detalladas a continuación:

- Administrador de la configuración: Supervisar que el equipo de desarrollo utilice el repositorio durante el ciclo de vida del proyecto. Controlará también que se respeten los lineamientos establecidos a la hora de evaluar el proceso de cambio.(*Cavanagh Juan*)
- Administrador de Versiones: Controlar que el producto cumpla con todos los requerimientos establecidos y su funcionamiento sea correcto.(*Casabella Martin*)
- Administrador del equipo de desarrollo: Interactuará con el repositorio operando sobre los ítems de configuración generados durante el proyecto. Principal consumidor de la información puesta bajo el control de configuraciones.(*Kleiner Matias*)

3. Administración de cambios

Se entiende como cambio todo aquello que afecte la línea base del sistema. Los cambios pueden proceder tanto a mejora como a la corrección de errores, el procedimiento para el manejo de cambios se realizará de la siguiente manera: cada vez que se realiza una solicitud de cambio es deber llenar un formulario, el cual debe ser entregado a los responsables para que procesen la solicitud, esta llegara a un estado final en el que será aceptada e implementada o postergada.

3.1 Solicitud de cambio

Los pasos necesarios para realizar una petición de cambio son:

1. Definir sector del software a modificar
2. Describir detalladamente los cambios a realizar
3. Proponer ciertos plazos para su implementación
4. Establecer un título al cambio requerido
5. Realizar una breve "experiencia de usuario" sobre el requisito planteado

Una vez cumplidos los pasos descritos anteriormente, se presentará al Administrador de Versiones, quien someterá dicha petición al proceso de cambio correspondiente (S 3.3)

3.2 Comité de Control de Cambios (CCC)

El comité de control de cambios estará integrado por un grupo de especialistas en Administración de proyectos, métricas, riesgos y requerimientos, cuya función será la de analizar el impacto y los costos de los cambios, y un aprobador, quien será el encargado de aprobar o no un cambio en base al análisis del grupo de analistas.

Integrantes del CCC

- Administrador del proyecto: Realizará la planeación de las actividades del diseño del proyecto, reportará oportunamente los cambios que haya en cuanto a duración, planeación de costos y recursos.(*Cavanagh Juan*)
- Especificación de requerimientos: Planeará, identificará, analizará y proporcionará las bases para poder hacer cambios en el desarrollo e implementación de los requerimientos.(*Casabella Martin*)
- Administrador de riesgos: Identificará los posibles riesgos que se presentarán al aprobar los cambios solicitados. Y presentará los posibles costos que llevará a cabo el cambio.(*Kleiner Matias*)
- Administrador de desarrolladores: Se encargará de todo lo relacionado al desarrollo de la aplicación aportando estimaciones más reales y los posibles productos que se afectaría si el cambio es aprobado.(*Cavanagh Juan*)
- Aprobador: Se encargará de aprobar o no un cambio con base al análisis del resto del equipo. (*Casabella Martin*)

Reuniones: Se realizarán reuniones semanales para evaluar el estado actual del proyecto, cada uno de los miembros presentará sus peticiones de cambio. Se contactará a los miembros del comité vía mail y, en caso de emergencias se llamará por teléfono. Será de gran importancia la presencia de los tres miembros de la CCC en todas las reuniones.

3.3 Proceso de control de cambios

Una vez presentada la propuesta de cambio, cumpliendo lo establecido en el punto 3.1, dicha propuesta se someterá al siguiente proceso para su correcta evaluación:

1. Definir si la información recibida es suficiente para la evaluación global del cambio.
2. Identificar las áreas afectadas por el cambio requerido.
3. Detallar costos estimados, fechas de apropiación y estimación de la implementación.
4. Convocar al CCC para el análisis económico y evaluación integro del cambio requerido.
5. Si es aceptado por CCC, se lleva a cabo el desarrollo del cambio a través de un documento escrito para los desarrolladores.
6. Si no es aceptado por CCC, se promueve el cambio al archivo de requerimientos pendientes.

4. Normas de nombramiento de archivos

El proyecto será guardado bajo el directorio (repositorio de github).

En ella se encuentran las siguientes subcarpetas:

- *src:* guarda el contenido de la aplicación (código fuente) y los test unitarios
- *doc:* guarda datos importantes del proyecto: plan de configuración, diagramas (en subdirectorios "DiagramasUML"), archivo de requerimientos y subdirectorio ejecutables (con el archivo .zip de los ejecutables del software)
- *gradle:* guarda la configuración del builder.
- *config:* guarda el checkstyle del proyecto
- *bin:* guarda documentos compilados del proyecto
- *cambios:* Contiene tres subdirectorios que son CambiosDescartados, CambiosPendientes, CambiosAprobados

A continuación se muestra una imagen de la organización del Directorio:

Directorio	Significado
src	Codigo fuente del proyecto
doc	Documentos asociados al proyecto
doc/Imágenes	Imágenes asociadas al proyecto
doc/Diagramas	Diagramas asociados al proyecto
doc/Ejecutables	Ejecutables asociados al proyecto
test	Archivos de testing del proyecto
cambios	Organización de los cambios
cambios/CambiosDescartados	Cambios que fueron descartados por la CCB
cambios/CambiosPendientes	Cambios pendientes a revision por la CCB
cambios/CambiosAprobados	Cambios aprobados por la CCB
bin	Uso de integracion continua, archivos compilados del proyecto

5 Equipos en BITSoftware

Equipos Scrum: encargados de desarrollar nuevas funcionalidades de forma ágil y rápida. Actualizan el código frecuentemente, evitando problemas a la hora de unir las partes desarrolladas por cada uno. También pueden estar encargados de corregir bugs en el programa.

Administración de etiquetas: Serán los encargados de establecer y verificar el cumplimiento de las normas de etiquetado. Deberán verificar que estos releases cumplan con los estándares de calidad establecidos en el proyecto

Equipos de rápida reacción: Se encargan de solucionar los errores urgentes, los cuales los clientes no pueden esperar a la próxima versión para que sean corregidos. Este equipo también se dedica a tareas de desarrollo y corrección de errores en casos en los que se vean comprometidas las ramas en las que trabajan.

Equipo de documentación: deberán crear y mantener actualizada la documentación del producto que será entregada al cliente. Esta documentación ayudará al cliente a entender las funcionalidades del proyecto y su implementación. Por ejemplo: archivos README, Guía de usuario, respuestas a preguntas frecuentes, etc.

Equipos de nuevos desarrollos: El objetivo de este equipo será descubrir e identificar las diferentes aplicaciones y los módulos que se pueden integrar al proyecto. Trabajaran en repositorios ficticios hasta que se decida presentar la petición de cambio y esta se apruebe, entonces se añadirá el proyecto al repositorio principal.

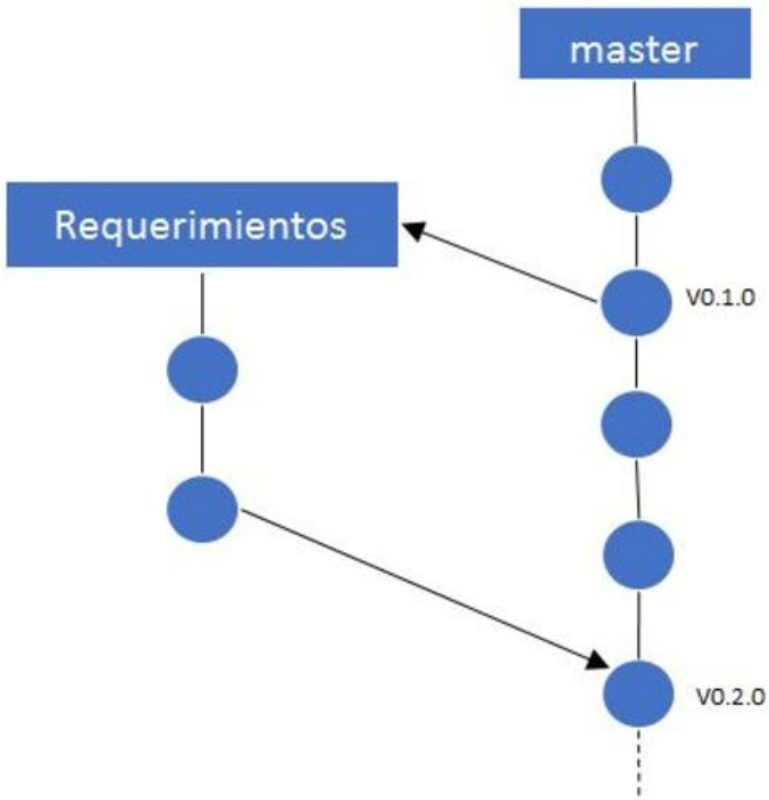
6 Administración del código fuente

En esta sección se describen distintos ítems de configuración. Se tiene en cuenta desde el esquema de ramas, el etiquetado, la estrategia de unión de archivos y el cumplimiento de los niveles de calidad para el producto final.

6.1 Esquema de ramas

- Rama de Función: La creación de rama se da lugar para la realización o modificación de una funcionalidad.
- Rama de Cliente: Si se aprueba la solicitud de un cliente, se creará una nueva rama (S 6.4)
- Rama Master: En esta rama se encontrá el desarrollo de software sin errores. Se podrá corregir en el mismo detalles que sean mínimos y no alteren el correcto funcionamiento del sistema. Se hará fusiones de archivos sobre esta rama (S 6.3)
- Rama de Modificación: Este tipo de rama será utilizada para realizar modificaciones en los documentos asociados al proyecto, tales como requerimientos, diagramas, datos históricos e incluso el plan de configuraciones. La identificación de la rama deberá corresponder al documento en cuestión. (Ej: branch Requerimientos; donde el archivo a modificar corresponde a los requerimientos del sistema).

A continuación se muestra una imagen con el esquema de ramas a seguir:



6.2 Definición de etiquetas

Se etiquetara un commit sólo si su código cuenta con correcta sintaxis y el codigo funcione correctamente. El etiquetado debe ser autorizado y revisado por los administradores de los equipos de desarrollo. También deberán ser respetadas las reglas que aquí se establecen para el nombramiento de la etiqueta. Las etiquetas estarán denominadas de la siguiente forma:

N1.N2.N3

- Primer dígito (N1): Este release indicará un gran cambio en la funcionalidad del proyecto agregando al menos dos características nuevas. Cada uno de estos releases deberá contar con autorización de un miembro de un mayor escalón jerárquico, ya que esta será una versión por la cual los clientes deberán pagar.
- Segundo dígito (N2): En este caso las versiones contarán con solamente una funcionalidad que difiere de la anterior, no estará disponible a los clientes a menos que los administradores indiquen lo contrario. El caso general será que se espere a algún otro release concretando así una versión más completa a brindarle al cliente.
- Tercer dígito (N3): Incluirá la corrección de bugs, cambios mínimos en el diseño de la interface que no afecten al funcionamiento. Esta versión será distribuida gratuitamente como actualización de la versión anterior que el cliente adquirió.

6.3 Estrategia de fusión de archivos

Dado el modelo de ramas que se decidió utilizar, notamos que una vez cumplido el fin con el que fue propuesta una rama, esta se da por finalizada y se una a la rama de integración (master) para ser implementada en el proyecto principal. Esta estrategia permitiría que dos equipos trabajan sobre el mismo proyecto en simultáneo sin molestar al otro. El problema recae a la hora de unir los ítems de configuración, frecuentemente se presentaran conflictos cuando se intente implementar el merge. Con la finalidad de resolver estos conflictos de la forma más rápida y eficiente posible se establecen una serie de reglas a la hora de implementar esta fusión de archivos:

- Se especificará la etiqueta de cierre (versión a la cual se quiere fusionar) y la etiqueta inicial (versión donde nace la rama).
- Se creará un archivo de texto .txt donde se especificará los detalles del estado actual de la fusión, cada desarrollador que haga un avance en este proceso deberá documentar los detalles aquí.
- Una vez terminada la fusión será revisada por el administrador de configuraciones para confirmar los cambios realizados en la rama de integración.
- Si los conflictos persisten, se acudirá al equipo de scrum, el cual puede decidir aislar algunos ítems en ambas ramas para lograr la correcta combinación.

6.4 Ramas por cliente

Para los cambios específicos de los clientes existirá una rama por cliente la cual evolucionará por su propia cuenta, y según los requerimientos de este cliente. Si estos requerimientos son demasiados, se creara un equipo de desarrolladores exclusivamente dedicados a este cliente.

7. Administración de builds

En el proyecto se utilizarán solo builds de integración continua. Estos se ejecutan cada vez que un desarrollador hace un commit, asegurando que este no rompa el código. Fuera ese el caso deja registrado quien fue el desarrollador que lo hizo y dónde está el error, además le envía un mail notificando de las fallas. En este casos se utilizará Travis CI, será el encargado de ejecutar de forma automática los test unitarios, test de sintaxis y generar reportes acerca de los resultados de los mismos. Informalmente también los desarrolladores realizarán builds a nivel local para corroborar que las funcionalidades nuevas que están aplicando son correctas, estos no se enviarán al repositorio, sino que sólo estarán disponibles en el ordenador de desarrollador.

8. Administración de Release

Se realizará un proceso para gestionar una nueva entrega o actualización de software.

8.1 Información y descripción

Se reúnen las nuevas exigencias, es decir la cumplimentación de todos los requerimientos. Se fijan pasos necesarios para la entrega, plazos límites, etc.

8.2 Release building

Una vez conocidos los requerimientos, dependencias y plazos, se comienza la construcción de la nueva entrega. Con todo lo que esto implica (diseño, codificación, cada parte de código, clase e ítem de configuración será compilado enlazado y armado en un ejecutable. Luego se realiza un tag en el repositorio para marcar la versión utilizada.

8.3 Pruebas de aceptación

Luego la build deberá pasar la prueba de aceptación, es decir que cumpla con requerimientos y que funcione correctamente para ser entregada al cliente.

8.4 Preparación de entrega

- Se documentan las fallas corregidas,
- nombre de entrega,
- especificación del entorno para el cual se ha construido la entrega,
- documentación,
- archivos de configuración,
- informe de pruebas.

8.5 Transferencia de release e instalación

Se transfiere al usuario la nueva release, de la forma pactada con anterioridad. Por último se realiza la instalación de la nueva entrega. a continuación tendremos a disposición la carpeta ejecutables, con cada versión del sistema, y su respectivo manual de instalación Ejecutables Manual de instalación

9. Backup

El código fuente del proyecto se encontrará alojado en los servidores de GitHub pero también existirá una versión de backup en un ordenador del proyecto, el cual copiará el proyecto de GitHub a diario para evitar que algún problema en el servidor afecte al correcto desarrollo del proyecto.