Plan de Gestión de las Configuraciones.

Ingeniería de Software.

08/05/2017

Integrantes

Índice.

Historial de revisiones.

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Fecha | Observaciones | Autor |
| 1.0.0 | 25/04/2017 | Armado inicial del documento. Introducción. | Monsierra, Lucas |
| 1.1.0\_DraftA | 29/04/2017 | Adición de características | Monsierra, Lucas |

Introducción.

# Propósito y alcance.

En este proyecto se desarrollará el Plan de Gestión de las Configuraciones de nuestro proyecto, en el cual se describirá la forma de trabajo, las políticas, los procesos y las herramientas que se usarán durante el desarrollo del software.

# Propósito del Plan de Gestión de las Configuraciones

A continuación se enumerarán alguno de los propósitos de éste:

* Garantizar consistencia en la información poniendo en práctica dicha gestión de las configuraciones.
* Definir las personar que participarán del soporte de la SCM.
* Mantener la integridad a lo largo de todo el ciclo de vida del producto.
* Asegurar que los cambios introducidos al sistema se hagan de forma controlada.
* Llevar registro con los detalles de dichos cambios.
* Informar el estado del proyecto.
* Rastrear las propuestas de cambio, almacenar versiones de componentes del sistema.
* Mejorar procesos.

# Referencias, Abreviaturas y Glosario.

* **SCM**: Gestión de la Configuración del Software (Software Configuration Management).
* **SVN:** Sistema de Control de Versiones (Version Control System).
* **CCB:** Comité de Control de Cambios (Change Control Board).
* **SCI:** Ítem de Configuración de Software (Software Configuration Item).
* **SCMer:** Rol encargado de realizar la tarea de gestión de la configuración (SCM).
* **CIT:** Herramienta de Integración Continua (Continuous Integration Tool).
* **BTS:** Herramienta de Seguimiento de Errores (Bug Tracking System).

# Conceptos básicos.

* **Repositorio:** Espacio físico donde se almacenan todos los elementos generados durante el proceso de desarrollo en sus diferentes versiones.
* **SCI:** Cualquier aspecto asociado al proyecto de software bajo control de configuración (diseño, código, datos de prueba, documento, etcétera).
* **Branch:** Creación de una nueva línea de código independiente a partir de una versión en una línea de código existente.
* **Merge:** Creación de una nueva versión de un componente de software al combinar versiones separadas en diferentes líneas de código.

Roles de la SCM.

# Administradores de la Gestión de Configuración.

La Administración de la configuración del proyecto está a cargo del Global Project Configuration Manager (GPCM), el cuál es asignado a una persona. Además se designa un backup GPCM, el cual también es un integrante del equipo.

|  |  |  |
| --- | --- | --- |
| **Rol** | **Primario** | **Backup** |
| GPCM | Monsierra, Lucas | Rao, Maximiliano |

# Responsabilidades de la Gestión de Configuración.

**GPCM:**

* Posee toda responsabilidad sobre todos los CI.
* Responsabilidad en la creación de branches y administración de sus políticas.
* Responsabilidad y asistencia sobre el etiquetado y lanzamiento de branches.
* Coordinar actividades del CM en el proyecto.
* Asegurar la correcta ejecución del esquema del CM.
* Participación en auditorias.
* Analizar todas las novedades relacionadas al CM.

**TPCM:**

* Asistencia en la creación de tags y branches.
* Creación de actividades para el equipo específico en branches.
* Garantizar la integridad del producto y el seguimiento de los elementos de configuración propios del equipo.
* Participación en auditorias.
* Analizar todas las novedades relacionadas al CM.

**Team:**

* Ayudar a resolver conflictos durante la actividad de merge.
* Asegurarse que los criterios de calidad de los entregables a la rama principal se cumplan.
* Seguir todos los procesos asociados, políticas y prácticas definidas por sus roles asignados.

Dirección y forma de acceso a las herramientas.

SVN.

GitHub - <https://github.com>.

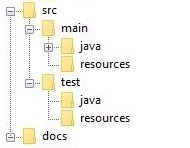
CIT.

Tavis CI - <https://travis-ci.org>.

# BTS.

Git Issues - <https://github.com/issues>.

Esquema de directorios y propósitos.



# Propósitos.

|  |  |
| --- | --- |
| **Ruta** | **Propósito** |
| .\src\main | Contiene las clases principales del proyecto. |
| .\src\test | Contiene las clases de los test usados en el proyecto. |
| .\docs\ | Contiene documentación y diagramas del proyecto. |

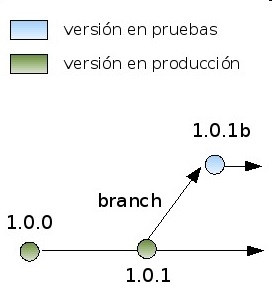
Normas de etiquetado y de nombramiento de los archivos.

Los identificadores usados seguirán una notación formada por el nombre del CI seguido de una notación numérica compuesta por tres números (y un cuarto opcional) separados por puntos, por ejemplo: “nombre”.V mayor.menor.revision[release].

Cada uno de estos componentes tiene un significado:

* Nombre del ítem de configuración.
* **Mayor:** indica la versión principal del software, la cual cumple con los requerimientos principales que se solicitan para dicha versión.
* **Menor:** indican funcionalidad menor cubierta en dicha versión.
* **Revision:** son las correcciones de bugs y fallos del IC.
* **Release:** lleva la cuenta de la cantidad de veces que se rechaza la entrega por incumplimiento de alguno de los requerimientos.
* Para el caso de la creación de un branch se indicará con la letra “b” al final de la numeración.

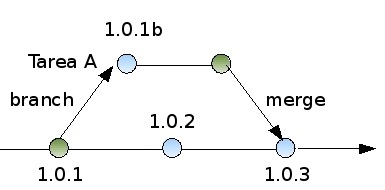
Plan del esquema de ramas a usar.

Como se aclaró anteriormente, un branch es una nueva línea de código independiente de la línea principal, pero creada a partir de esta. Generalmente se trabajara sobre la línea principal del proyecto, pero en ciertas ocasiones es necesario crear una línea de desarrollo paralela, para esto se utilizarán los branch.

Por ejemplo, supongamos que por el momento nuestro proyecto se está desarrollando solamente sobre una línea principal y algunas de las versiones se encuentran en entorno de producción. Pero se detecta un error crítico sobre la versión en producción, el cual hay que solucionarlo de manera rápida para que luego pasen el proceso de pruebas correctamente.

En esta situación se debe de crear un branch sobre dicha versión para trabajar en la solución de dicho error. El siguiente diagrama se ilustra este ejemplo.

Finalizada la corrección del bug, se tendrá una versión que, pasando el proceso de pruebas, se podrá poner en producción. Para evitar que se repitan tareas en las distintas líneas, cada branch debe representar el desarrollo una tarea concreta para que se vuelva a integrar en la línea principal de desarrollo en poco tiempo, de esta manera la tarea de integración será menos costosa.



Políticas de fusión de archivos y de etiquetado de acuerdo al progreso de calidad en los entregables.

La modalidad de fusión de archivos que se usará consiste en realizar un “rebase”, el administrador en conjunto con los desarrolladores, de esta manera se podrán identificar más rápido y solucionar los errores, donde el administrador estará siempre al tanto de lo que ocurre con los cambios.

Explicado de manera sencilla, el rebase es una herramienta para unir ramas, que reproduce los cambios que consolidamos uno a uno en nuestra rama de trabajo y los lleva a la rama a donde queremos unirlos, es decir, genera una especie archivos virtuales que vienen siendo nuevas consolidaciones en la rama a la que se le unirán los cambios y los ubica uno detrás del otro en el mismo orden que se realizaron.

También se definirán los criterios para incrementar los números de las etiquetas de versión:

* **Mayor:** se debe a la inclusión nuevas funcionalidades, claves de la aplicación, que pueden ser consecuencia de nuevos requerimientos para el sistema, como la inclusión de nuevos módulos o una revisión completa de los existentes.
* **Menor:** se deben a cambios significativos en la forma en la que se ofrece la funcionalidad existente, corrección de bugs del sistema o nuevas versiones evolutivas que modifican significativamente la funcionalidad ofrecida.
* **Revision:** se modifica por cada release de software que se realice.
* **Release:** se incremente al rechazarse una un release. Si la entrega se aceptase, se crea un tag público donde sólo se incluyen los tres primeros dígitos (mayor.menor.revision).

Forma de entrega de los “releases”, instrucciones mínimas de instalación y formato de entrega.

Luego de haber corrido los tests y encontrado que el nivel de fallas es aceptable, se procede a la compilación del release.

El compilador genera un archivo de extensión .apk, que se deberá transferir a un dispositivo móvil con sistema operativo Android, donde se instalará el software.

Éste, puede ser distribuido por medio de un sitio web, transferido entre dispositivos móviles o computadoras.

Change Control Board (CCB).

# Propósito.

Este grupo es responsable de tomar las decisiones sobre cómo debe evolucionar el sistema de software. Revisar y aprobar todas las peticiones de cambio considerando el efecto del cambio desde un punto de vista estratégico y organizacional más que técnico.

Decide si el cambio en cuestión está justificado económicamente y prioriza los cambios aceptados para su implementación, los cuales se transmiten de regreso al grupo de desarrollo y las peticiones de cambio rechazadas se cierran y ya no se emprenden más acciones.

# Formulario de pedido de cambio.

Cuando se quiere hacer un pedido de cambio, el mismo debe ser expresado en un formulario donde se deben incluir los siguientes ítems:

* Nombre del proyecto.
* Solicitante del cambio.
* Fecha.
* Cambio solicitado.
* Analizador del cambio.
* Componentes afectados.
* Prioridad del cambio.
* Esfuerzo estimado y real.
* Decisión.
* Persona responsable del cambio.
* Costo estimado y real.
* Pruebas y reportes.

El cuál será analizado por el CCB y dictará su decisión, como se explicó anteriormente.

El último ítem es muy importante, ya que si se realiza algún cambio, se deben realizar nuevamente las pruebas controlando que no se haya alterado alguna funcionalidad existente.

# ****Roles de los integrantes.****

Los roles que se desarrollarán en el CCB son los siguientes:

* Engineering Manager.
* Release Manager.
* Uber Scrum Master.
* GPCM.

Los cuales, al ser solamente 3 integrantes en el grupo, serán compartidos entre todos.

|  |  |
| --- | --- |
| **Miembro** | **Roles** |
| Benítez, Jeremías | Engineering Manager. |
| Rao, Maximiliano | Release Manager. |
| Monsierra, Lucas | GPCM, Uber Scrum Master. |

# Reuniones.

La organización del proyecto se basó en una reunión real donde se definió el proyecto a realizar y posteriores reuniones virtuales a través de un grupo de Whatsapp, donde se terminó de acordar los detalles del programa, las herramientas a utilizar y definir las actividades de cada uno.

Se determinará un día de la semana para llevar a cabo una reunión real, en la cual, el grupo se pondrá al tanto de todas las novedades en el proyecto. Además, ante cualquier eventualidad urgente, se realizarán comunicaciones a través de Whatsapp, y para el caso de desarrollo en conjunto también se utilizarán medios como Hangouts y Skype para las reuniones virtuales.

Herramienta de seguimiento de defectos usada para reportar los defectos descubiertos y su estado. Forma de acceso y dirección.

Como se expresó anteriormente, la BTS a utilizar será Issues, proporcionada por GitHub. Una vez creado el repositorio, se tiene acceso a ella.

La dirección y forma de acceso se encuentra definida en la parte superior.