Gastar Group

Reproductor de Música

Configuration Managment Plan (CM_Plan)

Autores:
<Gasparini, Roman;
Tarres, Martin>

Document Version: 1.0.1

Historial de Revisiones

Version	Fecha	Resumen de Cambios	
1.0.0	27/04/2017	Primera versión	

Página de aprobación

En la siguiente tabla se listan las personas encargas de cada area, y a las que se deberá acudir en caso de que se deba realizar algun cambio mayor.
La aprobacion no es necesaria si se trata de un cambio menor.
Sólo se realizan cuando el cambio es solicitado por una persona ajena al CM.

Configuration Management	Tarres, Martín

Software Developer	Gasparini, Roman

Tabla de contenidos

1 Introducción

1.1 Propósito y Alcance

Dentro del siguiente documento podemos encontrar descripciones detalladas sobre el Plan de Gestión de Configuraciones para el proyecto de un reproductor de música. El objetivo de este plan es controlar la configuración de los requerimientos, documentos, software y herramientas usadas en este proyecto.
Las herramientas se utilizaran para el control de versiones, para la integración continua, gestión de defectos, entre otros.
La gestión de las configuraciones de software (SCM) consiste en las herramientas y métodos utilizados para controlar el software a lo largo de su desarrollo y uso.

1.2. Propósito de Prácticas del SMP

- Garantizar la consistencia al poner en práctica las actividades de CM.
- Definir las autoridades que soporten las prácticas del CM.
- Mantener la integridad del producto a lo largo de su ciclo de vida.
- Informar a grupos e individuos interesados sobre el estado del proyecto.
- Crear un historial de los estados actual y anteriores de los productos desarrollados.
- Mejoras en el proceso de desarrollo.

1.3. Acrónimos

Acrónimo	Descripción
CM	Control de configuraciones
CBB	ChangeControl Board
SCM	Gestión de la configuración de Software
SMP	Plan de gestión de software

1.4. Herramientas del Plan de CM

Herramienta/Proceso	Propósito
Intellij IDEA	Entorno de desarrollo del proyecto
GitHub	Servicio de repositorio y hosting de ítems puestos en control de configuración.
Travis CI	Servicio de Integración Continua

2 Roles Gestión de Configuraciones

2.1. Configuration Manager del proyecto

Las actividades de gestión de las configuraciones dentro del proyecto “Reproductor de música” son coordinadas por el Manager Global del Proyecto de Configuraciones (GPCM), rol asignado a una persona.
El GPCM será responsable por actividades como controlar las ramas principales y las versiones de lanzamiento, determinando cuándo deben ser creadas las ramas y que actividades y características corresponden a cada rama.
Las actividades de manejo de configuraciones, procesos y sus respectivos procedimientos deben ser seguidas y respetadas por todos los miembros. Es la responsabilidad de cada persona seguir y aplicar el procedimiento de CM apropiado, de acuerdo con sus rol.

2.2 Responsabilidades en el Plan de Gestión de Configuraciones

Rol	Responsabilidades
Configuration Manager	Posee la responsabilidad global de todos los ítems en configuración. Responsable de aplicar etiquetas en los releases. Coordinar actividades de CM dentro del proyecto. Responsable de la creación de ramas y administrar las mismas. Asistir en actividades de merge y de construcción. Asegurarse de la integridad y rastreabilidad de los ítems en configuración del proyecto. Generar nuevo código con nuevas características periodicamente. Ayudar a resolver conflictos en las actividades de construccion y merge. Cumplir los requisitos de calidad en los distintos branches.
Software Developer	Seguir las practicas recomendadas en cada etapa del proyecto. Corregir bugs encontrados en el código. Asegurarse de la integridad del producto y su rastreabilidad a lo largo del tiempo.

3 Gestión de Cambios

3.1 Alcance

La gestión de cambios es un proceso que ocurre después de que la documentación y el código fuente están identificados y aprobados.
Los cambios incluyen modificaciones internas al enfoque documentado originalmente, debido a resultado de tests o solicitudes de integración de nuevas funciones.

3.2 Módulos de clientes y releases

3.2.1 CCB (Change Control Board)

La CCB es un comité que se asegura de que cada modificación es considerada apropiadamente, y autorizada antes de su implementación. La junta es responsable de aprobar, monitorear y controlar las solicitudes de cambios para establecer ramas de desarrollo de ítems de configuración. Las decisiones sobre las solicitudes de cambios son tomadas según el resultado de tests de calidad del producto.
Los integrantes de la junta se contactaran via email o por teléfono para concertar una reunión extraordinaria, o discutir cambios referidos al proyecto fuera de ellas.

3.2.1.1 Miembros

La siguiente tabla muestra los integrantes del equipo que asiste a las reuniones de la CBB.

Rol en la CBB	Nombre
Software development	Gasparini, Roman
Configuration Manager	Tarres, Martin

3.2.1.2 Frecuencia

Reunión de la CBB	Frecuencia
Reproductor de Música	2 veces por semana

3.2.1.3 Herramientas de gestión de cambios

Se usará un repositorio de Github para gestionar cambios y nuevas características del software a desarrollar.
URL: Repositorio GitHub

4 Identificacion de Configuraciones

El repositorio de Github será usado para guardar y compartir recursos de diferentes releases. Por cada Release, se generará un tag que identifique y permita separar una versión de otra.

5 Equipos de Desarrollo de Proyecto

El código fuente sera gestionado por los siguientes equipos

Release Management Team: está a cargo del testing necesario para que el producto pueda llegar a ser lanzado al consumidor, asegurando su calidad. Está a cargo de la mantención del repositorio y la gestión de los tests automáticos que se realizan luego de cada commit, así como también de crear y mantener la documentación que será entregada a los consumidores.

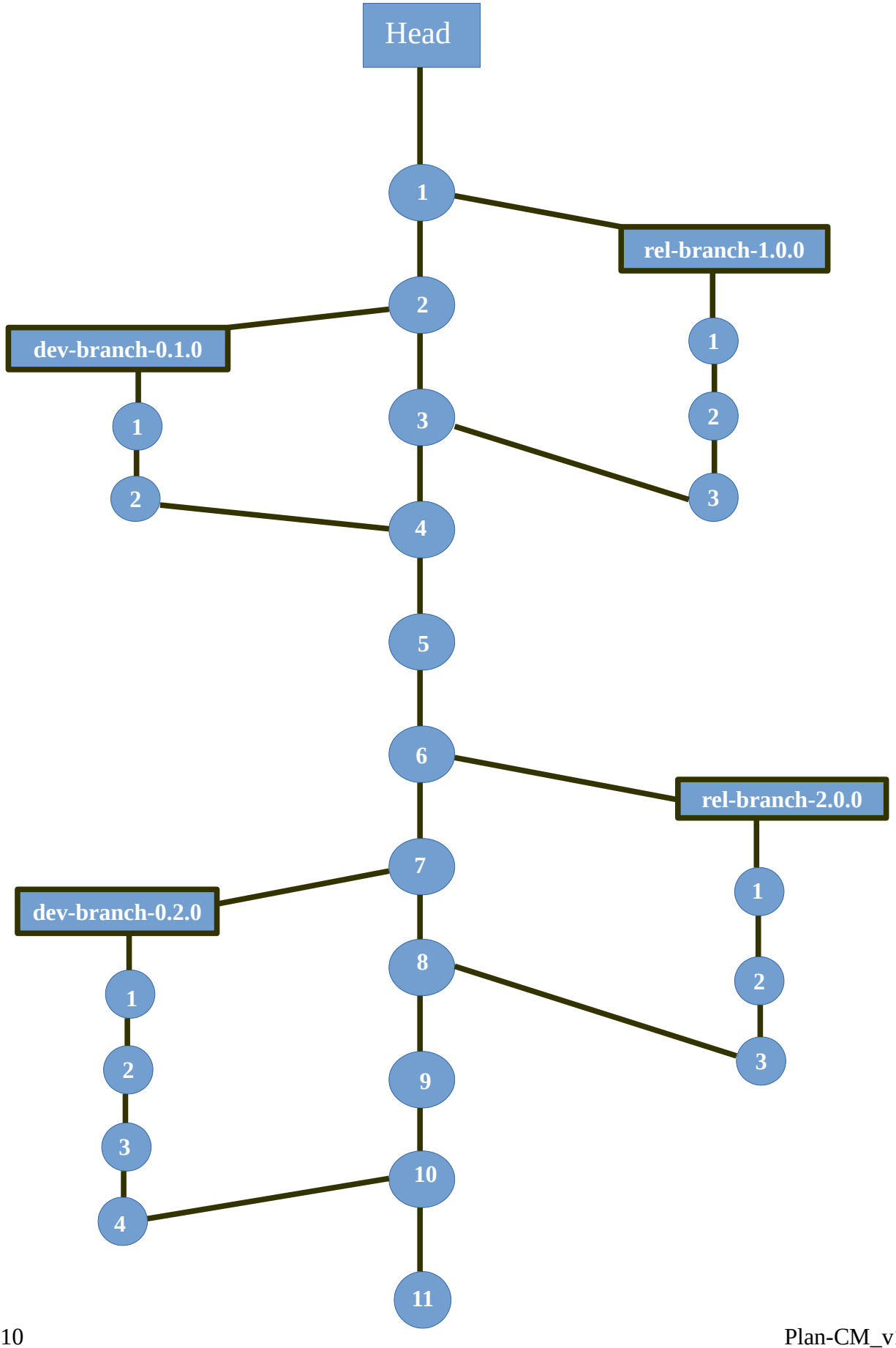
Development Team: a cargo de desarrollar nuevas funcionalidades, subir el código al branch correspondiente y hacer el merge a la rama adecuada según el calendario del proyecto. También pueden trabajar arreglando bugs o testeando características.

6 Gestión de la Configuración del Código Fuente

En esta sección se describen diferentes ítems de gestión de código fuente. Abarca aspectos referidos a esquema de ramificado, etiquetado, estrategia de fusión de código y niveles esperados de calidad para la totalidad del producto.

6.1. Módulos centrales

El gráfico siguiente muestra el esquema de ramificado que será seguido en los módulos centrales.


Head
1
rel-branch-1.0.0
2
1
dev-branch-0.1.0
3
2
1
2
3
4
5
6
rel-branch-2.0.0
7
1
dev-branch-0.2.0
8
2
1
2
3
9
3
10
4
11

6.1.1 Definición de Ramas

A continuación se definiran los tipos de ramificaciones que se utilizaran.

- **Rama Principal/Master:** es la rama donde se encontraran las vesrsiones estables y finales del producto, con la menor cantidad de bugs posibles y testeadas sus funcionalidades.
- **Development Branch:** en ésta se alojan las nuevas características, que antes de ser incorporadas al master deben aceitarse lo suficiente y deben testearse para alcanzar un nivel de calidad mínimo. La forma de nombrar estas ramas es dev-branch<cambio>.
- **Release Branch:** son ramas donde se integran características ya testeadas, y se apunta a una versión del producto para lanzar a los clientes. Se nombrarán como rel-branch<ID>.

Luego de definir los tipos de ramas, notamos que la principal motivación al utilizar este esquema es aislar el código inestable y bajo desarrollo del código estable y potencial a ser lanzado como producto final.
Es posible que se realicen desarrollos de nuevas características en la rama Master, sobre todo en una primera instancia, cuando el producto no tiene una estructura definida y no está cerca de un release inicial.
Además, pueden realizarse merges entre ramas de desarrollo y de release si así lo considera necesario el Configuration Manager, para luego integrar los cambios en el Master.

6.1.2 Definición de etiquetas (tags)

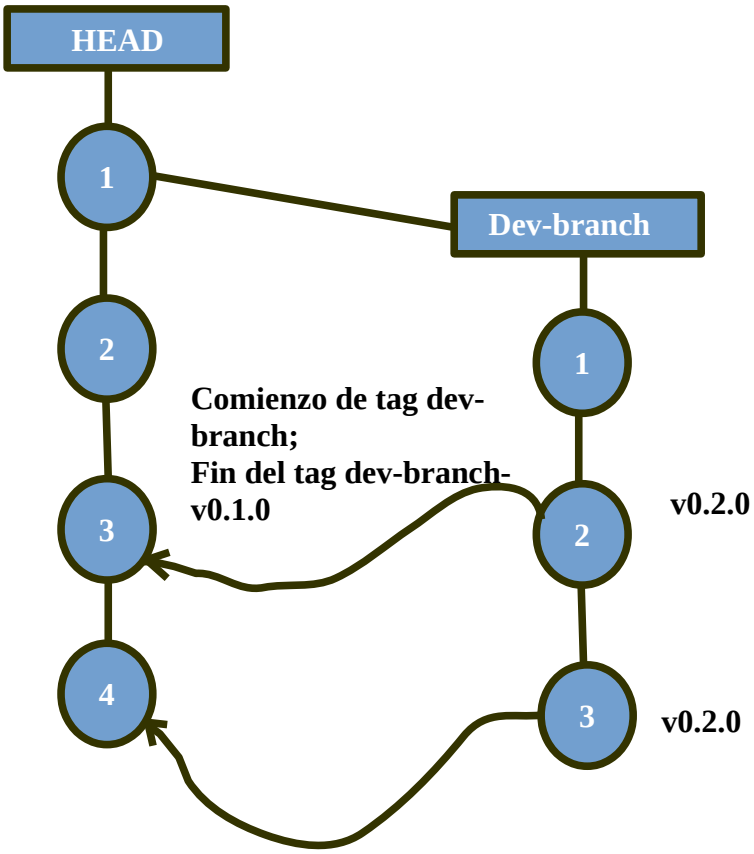
Release tags: estas etiquetas son aplicadas en las ramas de release. Son aplicados cuando se ha llegado a una versión final del release apuntado, finalizando su desarrollo en la rama correspondiente. El formato a seguir es el siguiente: rel-branch<ID>. Por ejemplo: rel-branch 1.0.0, rel-branch 1.2.0.

Development tags: en las ramas de desarrollo se etiquetara numéricamente la última versión del código, el que se considere apto para un merge hacia ramas más estables y de entrega de producto, para tener una referencia a la hora de realizar el merge y completar la documentación pertinente, siguiendo el esquema dev-branch<cambio><ID>.
El identificador representa las versiones del código puesto en configuración en esa rama, comenzando por el 0.1.0, luego el 0.2.0. Esta numeración será utilizada en la mayoría de los casos, cuando el nuevo contenido del archivo amerite un cambio de

nomenclatura.

6.1.3. Políticas de mergeo

Al realizar merges, lo mas recomendable es indicar el comienzo y final del tag, y por quien fue realizado.
Esto es útil para identificar las versiones afectadas por esta operación, en caso de ser necesario otro merge en esa misma rama, se hará partiendo del merge anterior.



Por ejemplo, en el gráfico se observa que se quiere hacer el merge entre la rama dev-branch y el HEAD, partiendo del tag dev-branch-singleton, y siendo el fin dev-branch 0.1.0.
Luego, se necesito realizar otro merge, siendo el nuevo tag dev-branch 0.1.0, y el nuevo tag dev-branch 0.2.0.
Aquí se puede apreciar que el nuevo comienzo de tag depende del ultimo fin tag utilizado, por lo que se realiza el merge partiendo desde el merge anterior, para evitar conflictos ya resueltos.
Se utilizará el archivo de notas de la versión Notas de Merge.txt, que irá evolucionando a la par del proyecto. Como información opcional, puede agregarse la lista de cambios del merge.

6.1.3 Archivos auxiliares CM

MergesNotes.txt:

7 Build Management

El tipo de Build a utilizar y que será implementado en todas las ramas es:
Continuous Integration Builds: Son útiles para identificar tan pronto como sea posible errores en el código y asi poder arreglarlos antes de hacer algún build formal. Están configurados para no aplicar ningún tag y para ser corridos automáticamente cada vez que alguien realice un commit a cualquiera de las ramas.
En la pestaña correspondiente de Travis CI se puede encontrar información útil, tal como el resumen de los tests empleados, el autor, la fecha del commit,etc.
La herramienta a utilizar para manejar el proceso de builds definido será Travis CI, que será debidamente configurada para enviar correos electrónicos a los grupos de trabajo que realicen el commit avisando de alguna falla en el build.

8 Release Management

Las Release builds serán realizadas en las ramas de desarrollo correspondiente. Estos serán enviados a los clientes por medio del repositorio de GitHub dándoles acceso al mismo para que descarguen los archivos necesarios, y puedan probar el proyecto en su IDE de Java de preferencia.

9 Backup y Recuperación de la Información

Los documentos y archivos del proyecto se guardaran en una carpeta del área de trabajo personal de cada uno de los integrantes, en caso de perder la información, cada uno lo puede proveer desde ahí. Este Backup se realiza al final de la jornada de cada trabajo.