

Juego de Ahorcado

Plan de Gestión de las Configuraciones

V1.1.0

07.06.2020

—

Grupo Random

Sebastián Klincovitzky

Cristian Pereyra

Ignacio Ruano

Santiago Moutón

Historial de Cambios

Versión	Fecha	Resumen	Autor
1.0.0	25-04-2020	Creación del archivo	Ignacio Ruano, Cristian Pereyra, Santiago Mouton, Klincovitzky Sebastian
1.1.0	07-06-2020	Corrección del CCB, Agregado de Nombramiento de Documentos	Ignacio Ruano, Cristian Pereyra, Santiago Mouton, Klincovitzky Sebastian

Tabla de contenidos

1.-Introducción	3
1.1 Propósito y Alcance	3
1.2 Acrónimos y Descripciones	3
1.3 Herramientas De Administración De Configuraciones	3
1.4 Roles y Responsabilidades	4
2.- Esquema de directorios	5
2.1 Estructura de Directorios y sus Propósitos	5
3.- Gestión de versiones	6
3.1 Normas de Etiquetado y Nombramiento de los Archivos	6
3.2 Guardando Cambios Históricos (Commits-tags)	6
3.3 Manejo de Almacenamiento	6
3.4 Desarrollo Independiente	7
4.- Gestión de la configuración del código	8
4.1 Esquema de Ramas	8
4.2 Política de Etiquetado de Ramas	8
4.3 Política de Fusión de Archivos	9
5. Gestion de entregas	10
5.1 Formato de Entrega de Releases	10
5.2 Instrucciones Mínimas de Instalación	10
6.- Gestión de Cambios	11
6.1 Introducción y Objetivos	11
6.2 Miembros	11
6.3 Frecuencia de Reunión de Trabajo	11
6.4 Proceso de Control de Cambios	11
6.5 Herramienta de Gestión de Cambios	12
7.- Herramienta de seguimiento de defectos	13

1. Introducción

1.1 Propósito y alcance del plan

Este documento abarca todo lo que se refiere al Plan de Gestión de las Configuraciones para un juego de “ahorcado”. El propósito del Plan de Gestión de las Configuraciones (PGC) es dirigir la estructura de los requerimientos, documentos, software y herramientas usadas para este proyecto.

1.2 Acrónimos y sus Significados

Acrónimo	Significado
PGC	Plan de Gestión de las Configuraciones
UML	Unified Modeling Language (Lenguaje Unificado de Modelado)
IDE	Integrated Development Environment (Entorno de Desarrollo Integrado)
CRF	Change Request Form (Formulario de Solicitud de Cambios)
CCB	Change Control Board (Junta de Control de Cambios)

1.3 Herramientas de Administración de Configuraciones

Herramienta	Prósito	Descripción
GitHub	Repositorio remoto para el control de versiones	Repositorio para el proyecto, permite la implementación de ramas.
Jenkins	Servidor de construcción para integración continua	Realiza la generación manual de compilación y testeo
LucidChart	Diagramas y diseños de arquitecturas (UML)	Herramienta para diagramas de arquitectura
Eclipse	IDE para codificación en Java	Desarrollo del código por parte de los integrantes del grupo en su workspace privado
Issues (GitHub)	Herramienta de gestión de cambio y defectos	Provista por GitHub permite realizar un seguimiento de consultas, informar sobre errores, entre otras características
Projects (GitHub)	Herramienta para asignación de tareas	Provista por GitHub, permite la asignación y seguimiento de tareas a realizar en el proyecto.

1.4 Roles y Responsabilidades

Se detalla a continuación, los roles y responsabilidades que serán comunes para todos los integrantes del proyecto:

- Definición del proyecto y de la asignación de recursos al mismo.
- Codificación mediante el correcto uso del repositorio local y común.
- Diseño e implementación de pruebas automáticas para el código trabajado individualmente.
- Resolución de problemas especificados en *Issues*.
- Definir y mantener el código fuente de uno o varios componentes, garantizando que cada componente implemente la funcionalidad correcta.
- Garantizar que se cumplan los requerimientos funcionales establecidos para el producto.
- Aprobación de resultados generales.
- Comprobación de calidad final del proyecto.

Lo siguiente establece qué funciones serán de responsabilidad individual:

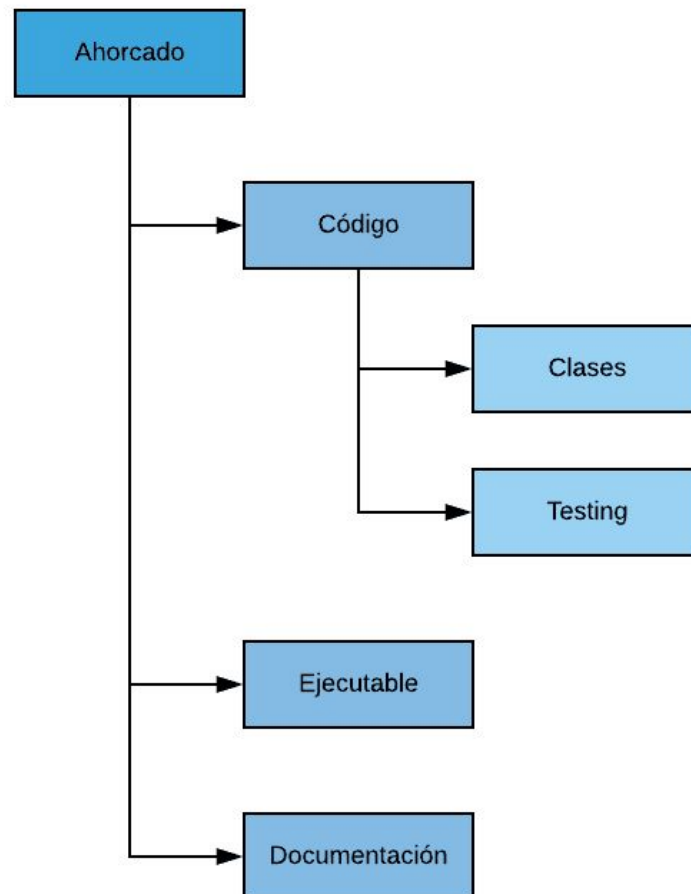
Integrante	Responsabilidad
Cristian Pereyra	Construcción de interfaz gráfica.
Ignacio Ruano	Control del etiquetado de versiones.
Santiago Moutón	Asiste en actividades de fusión de ramas.
Sebastián Klinevitzky	Seguimiento del cliente. Comunicación y entregas en general.

2. Esquema de Directorios

2.1 Estructura de Directorios y sus Propósitos

Nuestro esquema de directorio, consiste en un directorio raíz de nombre “Ahorcado”. Este contiene todos los subdirectorios. Se detalla a continuación el contenido del mismo:

- **Código:** Contiene la codificación del proyecto en lenguaje JAVA.. Además este contiene las siguientes carpetas:
 - **Clases:** posee el código de todas las clases definidas en java.
 - **Testing:** posee todo lo relevante al Testing del proyecto.
- **Documentación:** Contiene los documentos de Requerimientos y Configuración de Control
- **Ejecutable:** Contiene el ejecutable del proyecto



3. Gestión de versiones

3.1 Normas de Etiquetado y Nombramiento de los Archivos y Documentos

En el manejo de versiones se asignan identificadores basados en el nombre del ítem de configuración seguido por uno o más números.

El etiquetado y nombramiento de archivos y versiones va a seguir el estándar SemVer (Semantic Versioning 2.0.0) como guía, el cual propone un formato de versión X.Y.Z y donde X indica la versión principal, Y la versión secundaria y Z la versión de ajuste. Se pueden aclarar las siguientes reglas:


- X, Y y Z DEBEN ser números enteros, no negativos y mayores que 0.
 - Excepto X que puede ser igual a 0 solo para el desarrollo inicial.
- Una vez que una versión es publicada, los contenidos de esa versión NO DEBEN ser modificados. Cualquier modificación DEBE ser publicada como otra versión.
- El número de la versión principal (X.y.z) es incrementado al haber cumplido con todas las funcionalidades mínimas establecidas para una nueva entrega.
- El número de versión secundaria (x.Y.z) es incrementado al agregar funcionalidad nueva al programa.
- El número de versión de ajuste (x.y.Z) es incrementado al corregir algún comportamiento no deseado del programa.

Para los documentos se utilizará el mismo estándar de nombramiento que para los archivos, siguiendo las siguientes normas:

- El número de la versión principal (X.y.z) es incrementado al haber cumplido con todos los requerimientos y correcciones necesarios para la entrega.
- El número de versión secundaria (x.Y.z) es incrementado al agregar nueva información al archivo (ej. cosas que faltaron en versiones anteriores, gráficos, etc...)
- El número de versión de ajuste (x.y.Z) es incrementado al corregir secciones del documentos que ya existían. Ya sea por actualizaciones, por haberse encontrado información que no correspondía, ó por haberse agregado información faltante que se presupone.

3.2 Guardando cambios históricos (Commits-tags)

Todos los cambios hechos en el código del sistema son guardados y enlistados en un repositorio externo (ver **Herramientas de Administración de Configuraciones**). Estos cambios pueden ser usados



para seleccionar una versión del sistema en particular. Esto involucra el etiquetado (tagging) de componentes con palabras claves describiendo el cambio hecho (commits).

3.3 Manejo de almacenamiento

Para reducir el espacio de almacenamiento requerido para múltiples versiones, el sistema de gestión de versiones provee un gestor de almacenamiento que enlista la diferencia (deltas) entre una versión y otra. (ver **Herramientas de Administración de Configuraciones**)

3.4 Desarrollo independiente

Diferentes desarrolladores pueden estar trabajando en el mismo componente al mismo tiempo. El sistema de gestión de versiones hará un chequeo asegurándose que los cambios hechos no interfieran con los de otros desarrolladores. Para conocer cómo serán implementados estos chequeos, ver **Gestión de la Configuración del Código** en el apartado **Esquema de ramas**.

4. Gestión de la Configuración del Código

4.1 Esquema de ramas

El esquema consiste en cuatro tipos de rama, nombradas a continuación en grado de mayor a menor estabilidad:

- 1) **Master:** contiene las versiones del programa que se consideran lista para la entrega.
- 2) **Integración:** considerada una rama “pseudo-master”. Contiene las versiones del programa a las que se le ha añadido un funcionamiento nuevo de forma exitosa.
- 3) **Transitoria:** rama utilizada para la integración y resolución de conflictos de nuevas funciones.
- 4) **Ramas de Tarea:** ramas individuales para las distintas funcionalidades del programa (ej. interfaz, generación de palabras, etc.)

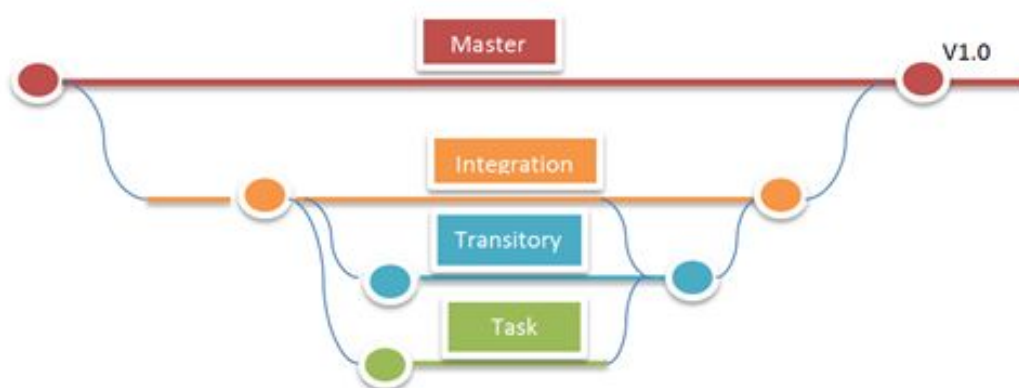



Fig. 1 - Diagrama gráfico del esquema de ramas

4.2 Política de Etiquetado de Ramas

Durante el desarrollo del programa, se trabajarán sobre las ramas indicadas previamente. A medida que los avances realizados sobre dicha rama pueda verse identificado con una nueva versión según la política establecida en Normas de Etiquetado y Nombramiento de los Archivos y Documentos, esta será subida al repositorio elegido.

Cada vez que se suba una nueva versión, a esta se le colocará una etiqueta. Esta etiqueta seguirá las normas de etiquetado explicadas anteriormente, donde se le asignará a la nueva versión de la rama un número según la severidad del cambio.



Como dato adicional, en el repositorio se deberá colocar un comentario indicando qué cambios fueron realizados a la rama.

4.3 Política de Fusión de Ramas

1. Primero, se trabajará sobre las ramas de tareas hasta haber completado un funcionamiento mínimo sin fallas aparentes.
2. Luego se copiará la rama Integración a una rama transitoria, y posteriormente se fusiona con la rama de una tarea, aquí se arreglará cualquier bug o conflicto que pudiera tener.
3. Posteriormente se fusiona con la rama transitoria con la rama Integración, entendiendo que ya se encuentra un programa estable. Esta rama funcionara como una rama pseudo-master.
4. En caso de tener ya una rama Integración que cumpla con funciones mínimas, o que el cliente pida una entrega de lo último, se hará una fusión con la master, originando una nueva versión disponible para release.

5. Gestión de Entregas

Una *entrega* es una versión del sistema de software que se le ha sido distribuido a los clientes. Estas versiones son posible de identificar gracias al etiquetado realizado utilizando la Gestión de Versiones descrito anteriormente.

Las entregas realizadas deben incluir:

- Archivos de configuración definiendo cómo la entrega debería configurarse para una instalación particular.
- Archivo de instalación (o ejecución) que es usado para ayudar a instalar el sistema en nuestro dispositivo.
- Archivo describiendo al sistema (manual de usuario).

El código ejecutable de los programas y toda la documentación asociada deberá estar correctamente identificada en el sistema de Gestión de Versiones y etiquetada con el identificador de la entrega.

5.1 Formato de Entrega de Releases

La compilación del Release se realizará en la rama “Master” del repositorio, donde por la forma de trabajo elegida con las ramas, contendrá una versión apta para ello. Para su entrega se le indicará al cliente que puede descargar en la página de GitHub la última versión comprimida en formato .zip.

5.2 Instrucciones Mínimas de Instalación

Se incluirá un archivo readme.txt en la carpeta a descargar en github que contendra la informacion para la instalación.

6. Gestión de Cambios

6.1 Introducción y Objetivos

Es un comité que garantiza que cada cambio sea considerado adecuadamente por todas las partes y que esté autorizado antes de su implementación. El CCB es responsable de aprobar, supervisar y controlar las solicitudes de cambio para establecer líneas base de elementos de configuración.

El objetivo es agregar, replantear o eliminar funciones, requisitos o requerimientos; para lograr un mejor comportamiento a nivel gráfico, lógico o de utilidad, como así también para corregir errores.

6.2 Miembros

- Cristian Pereyra
- Ignacio Ruano
- Santiago Moutón
- Sebastián Klincovitzky

6.3 Frecuencia de Reunión de Trabajo

Habrá reunión de trabajo una vez por semana, y en caso de que sea un cambio de urgencia se presentará en una oportunidad posible de cualquier día.

6.4 Proceso de Control de Cambios

El proceso de gestión de cambios es iniciado cuando un cliente completa y envía un pedido de cambio describiendo el cambio requerido en el sistema. Esto podría ser un reporte de error, donde los síntomas del error son descritos, o un pedido para el agregado de nueva funcionalidad en el sistema.

En etapas más avanzadas del proyecto, existe la posibilidad que por errores o tiempos sea necesario una modificación en el producto que será entregado con respecto a los planes originales. Estos cambios deben ser aprobados por la CCB.

Los cambios propuestos pueden ser presentados usando un formulario de solicitud de cambio (CRF). Cuando el cambio solicitado es procesado, la información es agregada al CRF guardando la decisiones hechas en cada etapa del proceso.

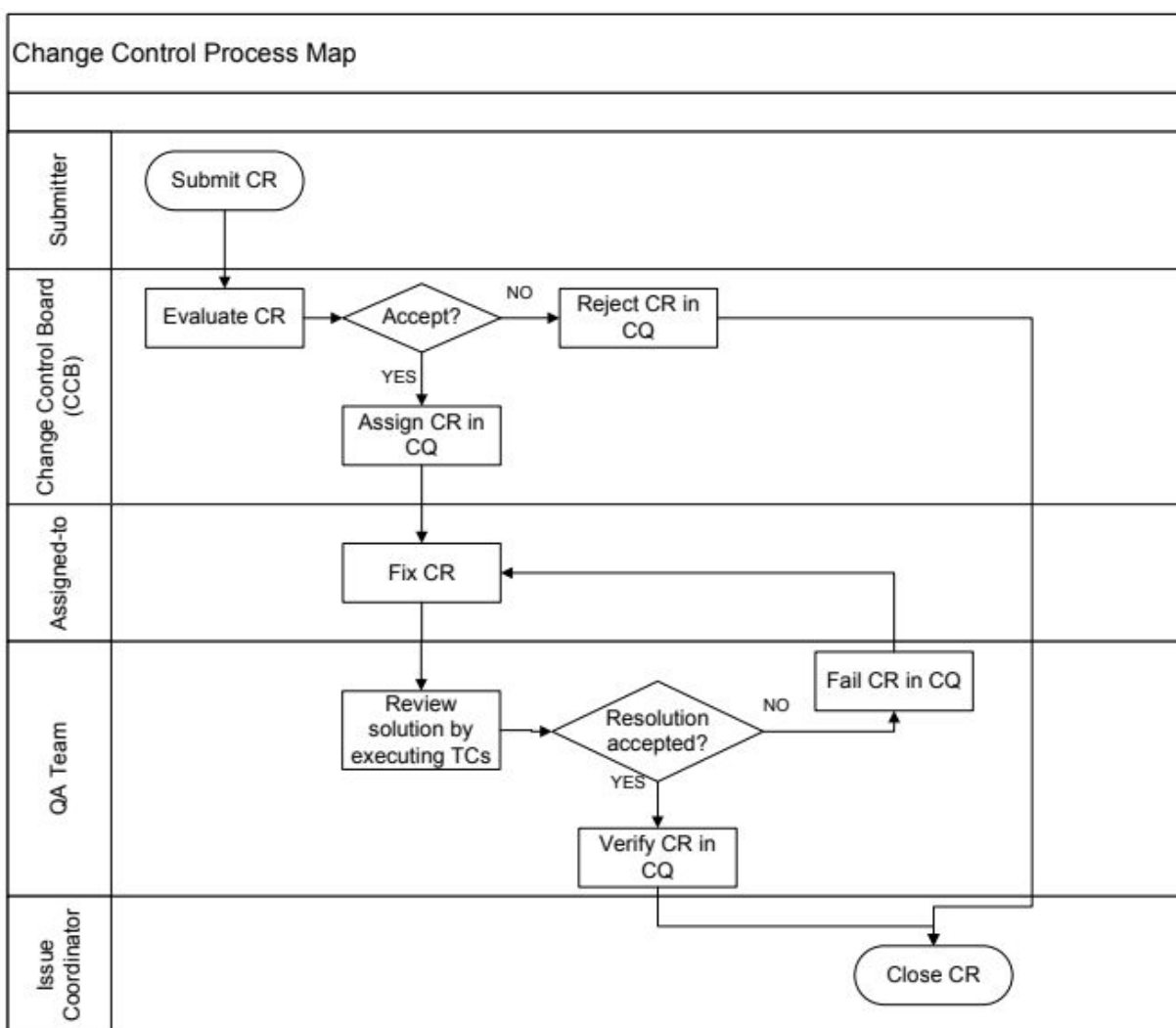
La CRF tendrá:

- Recomendaciones sobre el cambio en cuestión.
- El costo estimado del cambio.
- Los datos de cuando el cambio ha sido solicitados, aprobado, implementado, y validado.
- No hará énfasis en cuestiones de implementación.

Cada cambio propuesto por los miembros o el cliente deberán ser presentados primero, para luego ser aprobados o rechazados por el comité. Estos incluyen cambios en los planes, documentos y códigos.

El criterio de la gestión de cambios tomará en cuenta la calidad del producto, los beneficios, los costos, la dificultad que presenta, los impactos en los planes y las consecuencias que se presentarán por no aprobar los cambios.

A continuación se describe el proceso de control de cambios



Etapas	Involucrados	Descripción
Submitter	Cliente/Usuario	A través de la CFR, el cliente/usuario emite un pedido de cambios describiéndolo
CCB	Ir apartado Miembros	Se evalúa el pedido de cambio o aviso de error y se decide si se acepta o no, dependiendo de calidad del cambio, los beneficios, los costos, la dificultad que presenta, los impactos en los planes y las consecuencias que se presentarán por no aprobar los cambios.
Assigned to Development Engineering	Ingenieros Desarrolladores	Se agrega la nueva funcionalidad o se arregla el error reportado junto con su documentación.
QA team	Equipo de control de calidad	Se evalúa si la implementación es aceptada. En Caso de no serlo, se vuelve a asignar al los Ingenieros desarrolladores.
Issues Coordinator	Cordinador de Cambios	Se le da cierre al pedido de cambio o reporte de error.

Como se muestra, existen distintos roles involucrados en el proceso. En este caso, los miembros de la CCB serán responsables de todos ellos (excepto el de Cliente/Usuario)

6.5 Herramienta de Gestión de Cambios

Se usará la sección de *issues* de la página del proyecto de GitHub. Cada pedido de cambio debe titulado con la idea general, ser etiquetado con el label “enhancement”, ser acompañado por una descripción y el motivo por el cual se pide el cambio.



7. Herramienta de Seguimiento de Defectos

La herramienta de gestión de defectos a utilizar es Issues de GitHub. Con esta herramienta podremos crear cuadros de conversación y realizar un seguimiento entre todos los integrantes del proyecto sobre distintos aspectos del trabajo. Por nuestra parte, esta herramienta se utilizará para consultas concisas sobre la codificación, pero principalmente, para dejar asentados defectos, y buscar una solución mediante la cooperación de los miembros.

Una de las principales ventajas además de ser gratuita es que esta herramienta está implementada en el propio Github, el repositorio que se utiliza en el proyecto, y no requiere de la instalación de programas adicionales.